UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

INJEÇÃO REMOTA DE CHAVES EM POSs

João Pedro Figueiredo da Cruz

Dissertação orientada pela Prof. Doutora Maria Dulce Pedroso Domingos

DISSERTAÇÃO

MESTRADO EM SEGURANÇA INFORMÁTICA

2015

Agradecimentos

À Professora Doutora Dulce Domingos por todo o apoio e conselhos prestados ao longo destes meses.

Aos colegas e amigos Eduardo Pereira, Margarida Estorninho e Valentim Oliveira por todo o conhecimento partilhado.

Um obrigado muito especial a cada um de vós por todos os valiosos contributos. Sem eles, este trabalho não seria sequer possível.

Resumo

Points of Sale (POSs) são dispositivos móveis que permitem efetuar pagamentos de forma simples e eficiente utilizando um cartão bancário. Tipicamente, uma transação financeira nestes equipamentos resume-se à leitura do cartão (*chip* ou pista magnética) e à introdução do Personal Identification Number (PIN) para autorizar o pagamento.

No entanto, esta simplicidade encobre a complexidade inerente a uma transação, sobretudo no que concerne à segurança da informação do cliente que tem de ser transmitida desde o POS até ao processador da transação através de redes potencialmente inseguras (como a *Internet*). Esta informação é enviada cifrada com recurso a criptografia simétrica, onde o processador e o POS partilham uma chave secreta para cifrar e decifrar os dados sensíveis.

A forma de fazer chegar estas chaves secretas a ambas as partes constitui um momento fulcral na configuração inicial de um POS, pois se a injeção deste material criptográfico não for efetuada corretamente, estas chaves poderão ser descobertas e comprometer a segurança das transações feitas a partir desse POS.

Hoje em dia, em Portugal, estas chaves são injetadas de forma manual utilizando um de dois métodos: Os equipamentos têm de ser enviados ao processador para que este inicialize os terminais com estes segredos, ou o fabricante requisita um número de chaves secretas à mesma entidade e esta envia-as para que o próprio fabricante as insira nos seus POSs. Contudo, pela logística associada, estas soluções deixaram de ser viáveis pelo elevado número de equipamentos a inicializar atualmente.

Com este trabalho pretende-se investigar soluções que permitam agilizar este procedimento, de modo a fazer chegar esses elementos secretos a um POS remotamente, desde um ponto central de gestão de chaves criptográficas. São analisadas normas e outros documentos relacionados com o tema, cuja informação é útil para propor um protocolo de injeção remota de chaves seguro e eficaz, eliminando bastante intervenção humana no processo e diminuíndo a probabilidade de comprometimento das chaves criptográficas. A proposta é verificada formalmente quanto à sua segurança e também quanto à sua exequibilidade face a normas que o processador de transações tem, obrigatoriamente, de cumprir.

Palavras-chave: Criptografia; Segurança; Inicialização de POSs; Estabelecimento de chaves

Abstract

Points of Sale (POSs) are mobile devices used to make payments in a simple and effective way by using a banking card. Usually, a financial transaction on these terminals is executed by reading the card (its *chip* or magnetic stripe) and by introducing the Personal Identification Number (PIN) in order to allow the payment.

However, this simplicity hides the inherent complexity of a transaction, especially with regard to the security of customer information that has to be transmitted from the POS to the transactions processor through potentially insecure networks (like the Internet). This information is sent encrypted using symmetric encryption, where the processor and the POS share a secret key to encrypt and decrypt sensitive data.

The process to get these secret keys to be shared by both parties is a critical moment in the initial configuration of a POS because if the injection of this cryptographic material isn't performed correctly, these keys can be discovered and compromise the security of the transactions made from that POS.

Nowadays, in Portugal, these keys are manually injected using one of two methods: The devices has to be sent to the processor so that it initializes the terminals with these secrets, or the manufacturer requests a number of secret keys to the processor, which sends them to the manufacturer who inserts these keys in their POSs. However, because of the logistic associated with these procedures, these solutions are no longer practicable due to the high number of the devices currently being initialized.

The goal of this work is to investigate solutions to improve this procedure, so as to inject these secret elements to a POS remotely, from a centralized cryptographic key management facility. Some standards and other documents related to the subject are analyzed, which information is useful to propose a secure and effective remote key injection protocol, eliminating some human intervention in the process and decreasing the likelihood of compromise the cryptographic keys. The proposal is formally analyzed for its security as well as to its feasibility, comparing it to standards that have to be mandatorily followed by the transactions processor.

Keywords: Cryptography; Information Security; POSs initialization; Key establishment

Conteúdo

Lista de Figuras **xi**

Lista de Tabelas **xiii**

1 Introdução **1**
1.1 Motivação 1
1.2 Objetivo do trabalho 2
1.3 Resumo do trabalho realizado 2
1.4 Planeamento 3
1.5 Estrutura do documento 4

2 Trabalho relacionado **5**
2.1 Segurança de sistemas informáticos 5
2.2 Ataques à segurança da informação 6
2.2.1 Ataques passivos 6
2.2.2 Ataques ativos 6
2.2.3 Ataques de criptoanálise 7
2.3 Mecanismos de segurança 8
2.3.1 Chaves criptográficas 8
2.3.2 Criptografia simétrica 9
2.3.3 Criptografia assimétrica 9
2.3.4 Criptografia híbrida 10
2.3.5 Sínteses criptográficas 11
2.3.6 Message Authentication Codes (MACs) 11
2.3.7 Assinaturas digitais 11
2.3.8 Certificados digitais 12
2.3.9 Autoridade de Certificação 12
2.3.10 Frescura e ordem 14
2.4 Algoritmos criptográficos 14
2.4.1 Triple Data Encryption Standard (3DES) 14
2.4.2 Rivest-Shamir-Adleman (RSA) 17

2.4.3 Diffie-Hellman . . . 20
2.4.4 Algoritmos de síntese criptográfica . . . 21
2.4.5 Algoritmos de MAC . . . 21
2.5 *Scyther*: Analisador de protocolos de segurança . . . 23
2.5.1 Sintaxe do *Scyther* . . . 23
2.6 Conclusão do capítulo . . . 26

3 Enquadramento técnico **29**
3.1 Points of Sale (POSs) . . . 29
3.1.1 Principais características . . . 30
3.1.2 Chaves criptográficas . . . 31
3.2 Métodos existentes de injeção de chaves . . . 32
3.2.1 Carregamento local centralizado . . . 32
3.2.2 Carregamento local descentralizado . . . 32
3.3 Análise ao cenário de injeção de chaves atual . . . 33
3.4 Injeção remota de chaves . . . 34
3.4.1 A norma ANS X9.24-2 . . . 34
3.4.2 Proposta de implementação: ASC X9 TR34 . . . 36
3.5 Conclusão do capítulo . . . 43

4 Solução proposta **45**
4.1 Análise de requisitos . . . 45
4.1.1 Requisitos de criptografia simétrica . . . 45
4.1.2 Requisitos de criptografia assimétrica . . . 46
4.1.3 Requisitos do protocolo . . . 47
4.2 Entidades envolvidas . . . 48
4.3 Certificação de fabricantes . . . 48
4.4 Assinatura de certificado da CA do fabricante . . . 49
4.5 Geração de par de chaves de assinatura RSA . . . 50
4.6 Pré-instalação de par de chaves assimétricas num POS . . . 51
4.7 Protocolo de transporte de chaves simétricas . . . 52
4.7.1 Fase 1: Autenticação . . . 52
4.7.2 Fase 2: Transporte . . . 53
4.7.3 Fase 3: Confirmação . . . 55
4.8 Análise e resultados . . . 55
4.8.1 Análise via *Scyther* . . . 56
4.8.2 Avaliação via ASC X9 TR39 . . . 58
4.8.3 Detalhes de segurança . . . 60
4.9 Conclusão do capítulo . . . 63

5 Conclusão **65**

Abreviaturas **67**

Bibliografia **71**

Lista de Figuras

1 Modelo de criptografia simétrica 9
2 Modelo de criptografia assimétrica 10
3 Modelo de assinatura/validação de assinatura digital 12
4 Exemplo de certificado digital X.509 e cadeias de certificação 13
5 Algoritmo de cifra 3DES 15
6 Algoritmo de decifra 3DES 16
7 Diagrama de cifra e decifra RSA 19

8 Exemplos de vários tipos de POSs 29
9 Esquema de protocolo de transporte unilateral de chaves 35
10 Esquema de protocolo de transporte bilateral de chaves 36
11 Protocolo TR34: Fase de vinculação 37
12 *Key Block* TR34 38
13 Protocolo TR34: Fase de transporte 39
14 Protocolo TR34: Fase de confirmação 40
15 Análise ao Protocolo TR34 (*Scyther*) 41
16 Ataque ao Protocolo TR34 42

17 Modelo de confiança entre o KDH e os fabricantes de POSs 49
18 Estabelecimento da relação de confiança entre um fabricante, F, e o KDH 50
19 Estabelecimento de par de chaves de assinatura RSA, único por fabricante 51
20 Pré-instalação de elementos assimétricos em P 52
21 Fase 1 do protocolo proposto: Autenticação 53
22 Fase 2 do protocolo proposto: Transporte 54
23 Fase 3 do protocolo proposto: Confirmação 55
24 Resultados da análise à solução proposta (*Scyther*) 57
25 Ataque de *Signature Stripping*, se assinaturas não estiverem cifradas . . . 62

Lista de Tabelas

1 Planeamento inicial 3
2 Fases da dissertação 4
3 Ataques de criptoanálise 8
4 Avaliação do protocolo proposto via ASC X9 TR39 59

Capítulo 1

Introdução

Este capítulo pretende introduzir o trabalho realizado apresentando a motivação para o mesmo, os seus objetivos e o resumo do que foi efetuado no seu decurso. Adicionalmente, faz uma análise crítica à forma como decorreu o trabalho, comparando a sua execução com o pleaneamento apresentado no relatório preliminar.

1.1 Motivação

Todos os dias, milhões de transações financeiras são executadas utilizando variados mecanismos, sendo que entre os mais utilizados estão os Points of Sale (POSs) - terminais de pagamento que facilmente se encontram em qualquer estabelecimento comercial. Para o seu correto funcionamento, os POSs necessitam de estar ligados à entidade processadora de transações (ou, simplesmente, processador) de modo a transmitirem-lhe informação que lhe permita validar a exequibilidade da transação tal como o Personal Identification Number (PIN) do cliente, o número do cartão, entre outros dados. No entanto essa ligação poderá atravessar redes potencialmente inseguras (como é o caso da *Internet*) e, tendo em conta a informação sensível atrás mencionada, a exposição desses dados afetaria negativamente o cliente, o processador da transação e o comerciante envolvido.

De modo a proteger esta informação sensível, um POS partilha com o processador um conjunto de chaves secretas para cifrar e autenticar os dados de modo a estes poderem ser transmitidos de forma segura entre ambos sem pôr em causa a confidencialidade, integridade e autenticidade dos mesmos. Assim, torna-se vital garantir que estes segredos criptográficos são apenas partilhados entre as duas partes, caso contrário todos os dados cifrados por estas chaves seriam comprometidos e potencialmente usados em atividades fraudulentas. Por esta razão, atualmente o método que existe para que o processador e um dado POS acordem uma ou mais chaves secretas entre si passa por uma injeção manual deste material criptográfico no terminal: O processador gera as chaves, guardando-as de forma segura, e para as fazer chegar ao POS, os seus fabricantes recebem do processador as chaves secretas e injetam-nas nos seus equipamentos antes de sairem de fábrica. Nou-

tros casos, os POSs têm mesmo de ser enviados desde o fabricante até ao processador para que este insira as respetivas chaves, também de forma manual, devolvendo-os novamente ao fabricante.

Admitindo que o processador gere as suas chaves de forma segura e que, no caso dos POSs, as chaves criptográficas estão guardadas num componente inviolável denominado de Trusted Platform Module (TPM), a confidencialidade das mesmas está assegurada ainda que um atacante tenha acesso físico ao terminal. Estes métodos de injeção atuais têm-se revelado seguros mas acarretam um elevado peso logístico, seja pelo envio dos terminais até ao processador e depois de volta à fabrica, ou pela complexidade inerente à segurança do transporte de um grande número de chaves secretas até ao fabricante que é feito, hoje em dia, utilizando métodos tradicionais, como por exemplo num CD enviado por correio.

Por tudo isto e porque o atual número de POSs e o seu crescimento[1] inviabilizam estes métodos de injeção manual no futuro, torna-se necessário evoluir esta metodologia para uma **injeção remota de chaves**, em que os POSs são colocados no local da sua utilização final desprovidos de qualquer chave previamente acordada entre este e o processador e, mediante um protocolo seguro, as chaves secretas são enviadas desde um ponto de gestão central até ao equipamento, atravessando redes sem qualquer tipo de controlo cujo tráfego pode ser facilmente escutado por um atacante.

1.2 Objetivo do trabalho

Este trabalho tem como objetivo a apresentação de um protocolo seguro de injeção remota de chaves em POSs, utilizando para tal um conjunto de técnicas e algoritmos criptográficos que, no final, contribuirão para um método de inicialização criptográfica destes terminais mais seguro e eficiente do que os atuais mecanismos utilizados para o efeito. O resultado é analisado formalmente com recurso a uma ferramenta de análise de protocolos de segurança denominada ***Scyther***[2] que provará, juntamente com outras análises efetuadas, que o método sugerido é seguro, ainda que um adversário que pretenda comprometer estes segredos consiga intercetar a troca de informação entre a entidade distribuidora de chaves secretas e o POS.

1.3 Resumo do trabalho realizado

O trabalho executado iniciou-se pela busca, investigação e análise do panorama atual no que concerne a normas e boas práticas a seguir caso se pretenda definir um protocolo de acordo de chaves secretas entre duas ou mais partes. Dos documentos analisados,

[1] atualmente, em Portugal, existem mais de 200 mil POSs a operar.

[2] `http://www.cs.ox.ac.uk/people/cas.cremers/scyther/`

três destacam-se e são essenciais para a execução deste trabalho, sendo detalhados mais adiante:

- **ANS X9.24-2**: Retail Financial Services. Symmetric Key Management - Using Asymmetric Techniques for the Distribution of Symmetric Keys [2]
- **ASC X9 TR34-2012**: Interoperable Method for Distribution of Symmetric Keys using Asymmetric Techniques: Part 1 - Using Factoring-Based Public Key Cryptography Unilateral Key Transport [4]
- **ASC X9 TR39-2009**: Retail Financial Services Compliance Guideline - Part 1: PIN Security and Key Management [3]

Com essa informação e após feita uma análise de requisitos, foi desenhado um protocolo de transporte de chaves secretas com recurso a técnicas de criptografia assimétrica, baseado nos dois primeiros documentos. A solução foi analisada formalmente com recurso ao *Scyther* que mostrou que o mecanismo sugerido preserva a confidencialidade, integridade e autenticidade das chaves trocadas e que, quando comparado com outros protocolos analisados na fase inicial do trabalho (nomeadamente a proposta feita no segundo documento), oferece um nível de segurança superior. Finalmente, foi feita uma avaliação do protocolo proposto tendo em conta os objetivos e boas práticas definidas no terceiro documento, cuja norma deve ser cumprida por qualquer entidade do ramo financeiro que deseje executar um protocolo de injeção remota de chaves simétricas nos seus POSs.

1.4 Planeamento

A Tabela 1 resume o planeamento previsto inicialmente.

Actividade	**Duração**
Contextualização do problema	1 semana
Análise à norma ANS X9.24-2	1 semana
Investigação e análise de outras normas	2 semanas
Escrita do relatório preliminar	2 semanas
Análise de Requisitos	3 semanas
Desenho do protocolo	3 semanas
Enquadramento do projeto a um fabricante de POSs	1 semana
Modificações à aplicação executada nos POSs	7 semanas
Implementação de um protótipo do protocolo	10 semanas
Análise ao protocolo desenvolvido	2 semanas
Otimização do protocolo desenvolvido	2 semanas
Prova de conceito e análise aos resultados	2 semanas
Escrita do relatório final	3 semanas

Tabela 1: Planeamento inicial

A Tabela 2 apresenta as fases pelas quais passou este trabalho e respetiva duração.

Actividade	**Duração**
Contextualização do problema	1 semana
Análise da norma ANS X9.24-2	1 semana
Investigação e análise de outras normas	2 semanas
Escrita do relatório preliminar	2 semanas
Análise de Requisitos	3 semanas
Compreensão e estudo dos atuais mecanismos de injeção de chaves	2 semanas
Análise e compreensão do documento ASC X9 TR34	4 semanas
Análise formal do protocolo de injeção remota de chaves TR34	2 semanas
Iterações do desenho do protocolo a propor	7 semanas
Análise formal à iteração final do protocolo	4 semanas
Avaliação do protocolo proposto perante normativo ASC X9 TR39	1 semana
Escrita do relatório final	5 semanas

Tabela 2: Fases da dissertação

Como se constata quando comparando ambos os planeamentos descritos, o trabalho foi alvo de uma restruturação significativa por razões alheias ao autor deste trabalho. Inicialmente foi planeada uma parceria com um fabricante nacional de POSs que apoiaria nos testes e numa prova de conceito do protocolo desenhado. Tal não foi possível e, derivado desse facto, optou-se por dar continuidade ao trabalho conduzindo-o por uma vertente mais formal, testando a proposta de protocolo com recurso ao *Scyther*, bem como com o auxílio do documento ASC X9 TR39, contendo as melhores práticas a ter em conta neste contexto.

1.5 Estrutura do documento

O documento divide-se nos seguinte capítulos:

Capítulo 1: Apresenta o problema a resolver e os objetivos deste trabalho.

Capítulo 2: Consolida conceitos úteis para compreender o trabalho desenvolvido.

Capítulo 3: Descreve as principais características dos POSs e ilustra o atual "estado da arte" no que concerne à inicialização criptográfica destes equipamentos, expondo ainda o conceito de injeção remota de chaves, bem como as bases que contribuem para o protocolo proposto neste documento.

Capítulo 4: Apresenta o protocolo de injeção remota de chaves desenvolvido no âmbito deste trabalho, mostrando também a análise à solução desenvolvida e respetivos resultados.

Capítulo 5: Apresenta a conclusão final deste trabalho.

Capítulo 2

Trabalho relacionado

Neste capítulo abordam-se vários conceitos no âmbito da segurança da informação e criptografia que estão relacionados com o trabalho realizado, tais como ataques e mecanismos de segurança, bem como alguns algoritmos criptográficos existentes. Introduz ainda a ferramenta analisadora formal de protocolos de segurança *Scyther*.

2.1 Segurança de sistemas informáticos

A segurança de sistemas pode ser definida da seguinte forma:

*"A proteção conferida a um sistema de informação por forma a atingir os objetivos de preservar a **confidencialidade**, a **integridade** e a **disponibilidade** dos recursos desse mesmo sistema, sejam eles hardware, software, firmware, dados ou comunicações."* [23]

Esta introdução faz uso de três conceitos considerados a base da segurança de sistemas informáticos:

- **Confidencialidade**: Assegura que informação privada ou confidencial não é divulgada a entidades não autorizadas. A perda de confidencialidade traduz-se na disponibilização não autorizada dos dados.
- **Integridade**: Previne alterações indevidas à informação. A perda de integridade traduz-se em alterações de dados por parte de entidades não autorizadas.
- **Disponibilidade**: Garante que o sistema está a trabalhar de acordo com o suposto e o serviço prestado não é negado a utilizadores autorizados. A perda de disponibilidade reflete-se em falhas no acesso aos dados ou ao sistema.

Contudo, estes não são os únicos conceitos a ter em conta na área da segurança de informação, sendo possível destacar outros dois igualmente relevantes:

- **Autenticação**: Assegura que a identidade dos participantes numa comunicação não é falsa e são precisamente quem estes afirmam ser. Pode ser considerada unilate-

ral (por exemplo, um servidor autenticar-se perante um cliente) ou mútua (quando cliente e servidor se autenticam entre si).

- **Não-repudiação**: Impossibilidade de qualquer interveniente numa comunicação negar a sua participação na mesma.

2.2 Ataques à segurança da informação

Os ataques à segurança de qualquer protocolo podem ser divididos em dois grandes grupos: Ataques passivos ou ativos [32].

2.2.1 Ataques passivos

Um ataque passivo visa a recolha de informação de um protocolo ou sistema não afetando o funcionamento do mesmo. Posteriormente, esta informação pode ser analisada pelo atacante de modo a perpetuar uma segunda fase do ataque. Duas formas de pôr em prática este tipo de ataque baseiam-se na **leitura de conteúdos** e na **análise de tráfego**.

Leitura de conteúdos

Um *email* ou um ficheiro transferido pode conter dados sensíveis que não deveriam ser lidos por ninguém não autorizado. Uma defesa contra estes acessos indevidos passa por cifrar a informação sensível de modo a que seja computacionalmente difícil decifrar os dados por parte de um adversário.

Análise de tráfego

Mesmo que dados sensíveis sejam transmitidos de forma cifrada existe ainda a possibilidade de uma atacante executar uma análise ao tráfego trocado entre duas entidades. Nessa análise, um adversário pode descobrir informações relevantes tais como a identidade e localização dos participantes na comunicação bem como o padrão e a frequência das mensagens trocadas. Esta informação pode ser útil para que o atacante entenda a natureza e o propósito da comunicação.

2.2.2 Ataques ativos

Por oposição aos ataques vistos atrás, os ataques ativos executam alterações e perturbações na troca de mensagens entre os participantes do protocolo podendo até induzir em erro um ou mais intervenientes fazendo com que estes troquem informação sensível com um atacante de forma não intencional. Estes ataques podem dividir-se nas seguintes categorias:

Personificação

Acontece quando um atacante se faz passar por outra entidade obtendo privilégios adicionais, personificando um dos intervenientes no protocolo que tenha esses mesmos privilégios.

Repetição

Envolve a captura de informação *à priori* (com recurso a um ataque passivo). Essa informação é, depois, reenviada para um dos intervenientes do protocolo de modo a que esta ação reproduza uma situação não autorizada.

Interposição

Envolve a interceção de mensagens entre dois interlocutores, podendo o atacante alterá-las indevidamente antes de as fazer chegar ao seu destino final. Este ataque é geralmente apelidado de "homem-no-meio".

Negação de serviço

Situação em que um sistema deixa de prestar o seu serviço normal após um atacante ter esgotado os recursos do sistema. Estes ataques podem ter como alvo algo mais específico, como por exemplo a supressão completa de mensagens que deveriam chegar a uma entidade.

2.2.3 Ataques de criptoanálise

Os ataques de criptoanálise baseiam-se no conhecimento do algoritmo de criptografia utilizado bem como no eventual conhecimento de pares de texto cifrado e texto em claro. O objetivo destes é o de deduzir, com base nessa informação, a chave secreta que cifra os dados em questão e, consequentemente, os dados em claro.

A Tabela 3 identifica os tipos de ataques de criptoanálise mais comuns, bem como o conhecimento que um criptoanalista tem em cada um dos casos [22]:

Tipo de ataque	**Conhecimento do criptoanalista**
Ciphertext-only	Algoritmo de cifra
	Dados cifrados
Known-plaintext	Algoritmo de cifra
	Dados cifrados
	Um ou mais pares de dados cifrados/decifrados
Chosen-ciphertext	Algoritmo de cifra
	Dados cifrados e os correspondentes dados em claro (que o sistema decifrou a pedido do criptoanalista, sendo que os dados inerentes aos pedidos são arbitrários)

Tipo de ataque	**Conhecimento do criptoanalista**
Chosen-plaintext	Algoritmo de cifra
	Dados em claro e os correspondentes dados cifrados (que o sistema cifrou a pedido do criptoanalista, sendo que os dados inerentes aos pedidos são arbitrários)
Adaptive chosen-ciphertext	Algoritmo de cifra
	Dados cifrados e os correspondentes dados em claro (que o sistema decifrou a pedido do criptoanalista, sendo que os pedidos não são arbitrários mas têm em conta resultados de pedidos anteriores)
Adaptive chosen-plaintext	Algoritmo de cifra
	Dados em claro e os correspondentes dados cifrados (que o sistema cifrou a pedido do criptoanalista, sendo que os pedidos não são arbitrários mas têm em conta resultados de pedidos anteriores)

Tabela 3: Ataques de criptoanálise

2.3 Mecanismos de segurança

De seguida apresentam-se mecanismos de segurança e criptografia utilizados na prevenção dos ataques vistos anteriormente.

2.3.1 Chaves criptográficas

Chaves criptográficas são componentes essenciais na criptografia. Muitos algoritmos criptográficos consistem em pares de operações, sendo o mais comum a cifra e decifra de informação. Uma chave é um conjunto variável de *bits* e alimenta um algoritmo criptográfico de modo a executar essas mesmas operações. Num algoritmo criptográfico desenhado corretamente a sua segurança depende diretamente da segurança da respetiva chave criptográfica e não de manter secretos os detalhes do algoritmo[1].

As chaves criptográficas podem dividir-se mediante a sua utilização num algoritmo criptográfico bem como pelo seu tempo médio de utilização. Estas podem ser **chaves simétricas** caso sejam utilizadas para ambas as situações de um par de operações criptográficas (i.e., para cifrar e decifrar). Podem também ser **chaves assimétricas** caso chaves diferentes sejam utilizadas para operações criptográficas distintas. O exemplo mais comum são os pares de chaves pública e privada, em que existe uma chave utilizada para cifrar informação e outra diferente para decifrar e obter os dados em claro.

Quanto ao seu tempo médio de vida, as chaves podem ser classificadas entre **chaves de longa duração** caso sejam utilizadas para operações criptográficas por um grande período de tempo ou **chaves de sessão** (ou de curta duração) se, por outro lado, forem utilizadas em operações de criptografia uma única vez, existindo uma baixa probabilidade de estas se repetirem num futuro próximo.

[1] Técnica denominada por "segurança por obscuridade".

2.3.2 Criptografia simétrica

Na criptografia simétrica, tanto o processo de cifra como de decifra de dados é efetuado **com base na mesma chave criptográfica**. Assim, caso um interveniente (*Alice*) queira comunicar de forma segura com outro (*Bob*), neste tipo de criptografia, *Alice* transforma uma mensagem em claro M numa mensagem cifrada C utilizando um algoritmo de cifra E e a chave secreta K: $C = E(K, M)$. O processo de reverter o conteúdo cifrado para a mensagem em claro por parte de *Bob* utiliza um algoritmo de decifra D e a mesma chave secreta utilizada para o processo de cifra: $M = D(K, C)$. A Figura 1 esquematiza este modelo de criptografia [33].

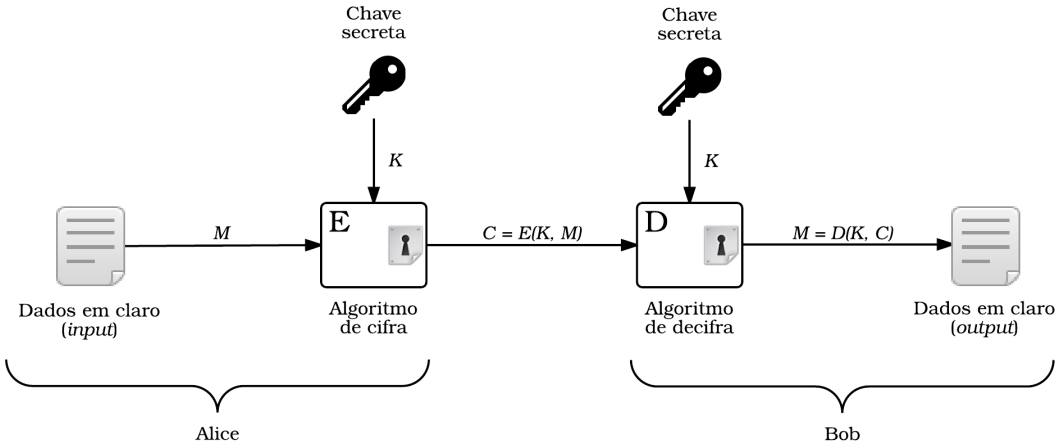

Figura 1: Modelo de criptografia simétrica

Neste contexto, a chave secreta K deverá ser obtida por parte dos emissores e recetores de mensagens de forma segura, sendo que os referidos interlocutores devem, impreterivelmente, garantir que esta se mantém secreta durante o seu tempo de vida, sob pena de um atacante conseguir decifrar toda a comunicação se a chave criptográfica for comprometida.

2.3.3 Criptografia assimétrica

Na criptografia assimétrica, ao contrário do caso da simétrica, o processo de cifra e de decifra de dados é efetuado **com base em duas chaves criptográficas distintas**. Destas duas chaves, uma delas é denominada **chave pública** e deverá ser do conhecimento de qualquer interveniente que deseje comunicar com o proprietário da chave. A outra denomina-se **chave privada** e, ao contrário da anterior, só deverá ser conhecida pelo seu titular.

O processo de cifra e decifra de dados num esquema de criptografia assimétrica está retratado na Figura 2 [33]. Novamente, *Alice* deseja enviar uma mensagem M a *Bob*. Para

isso, *Alice* requisita a chave pública de *Bob*, PU_{Bob}, de um repositório de chaves públicas previamente definido, cifrando os dados com a mesma: $C = E(PU_{Bob}, M)$. Por outro lado, após a receção dos dados cifrados, *Bob* consegue obter os dados em claro utilizando a sua chave privada PR_{Bob}: $M = D(PR_{Bob}, C)$. Nenhum outro terceiro interveniente poderá decifrar os dados pois apenas *Bob* tem conhecimento da sua chave privada.

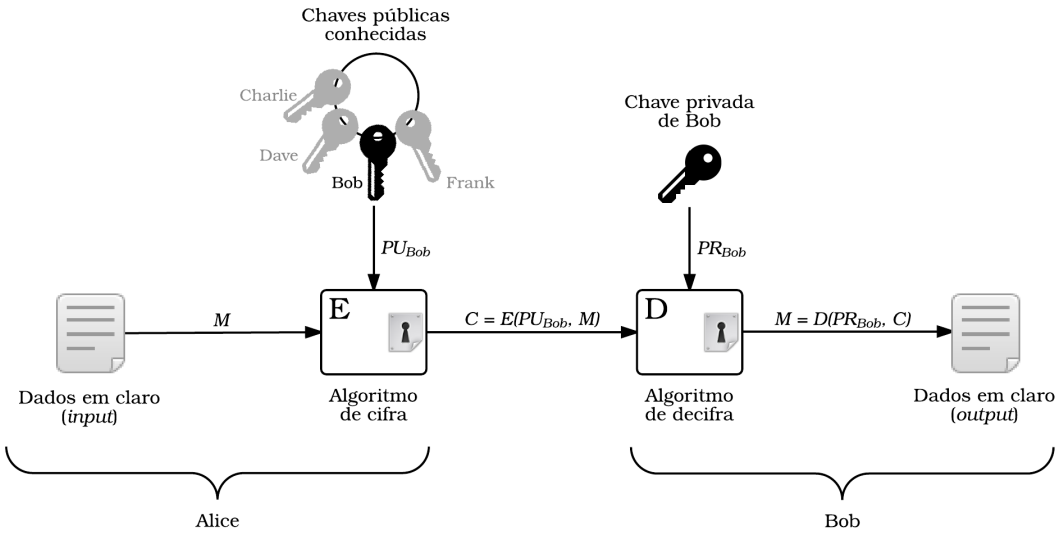

Figura 2: Modelo de criptografia assimétrica

2.3.4 Criptografia híbrida

Não obstante as diferenças existentes entre a criptografia simétrica e assimétrica, existe a possibilidade de combinar ambas de modo a formar o paradigma de criptografia híbrida. Este conceito alia a conveniência da partilha da chave pública para que o seu proprietário receba dados cifrados de forma segura com o melhor desempenho dos algoritmos de cifra simétrica, cujas operações de cifra e decifra são mais rápidas quando comparadas com as operações análogas da criptografia assimétrica [21]. Esta fusão é também conhecida pelo conceito de Key Encapsulation Mechanism - Data Encryption Mechanism (KEM-DEM) [14].

Assim, o objetivo da criptografia híbrida é o de transportar uma chave simétrica de sessão com recurso a um par de chaves assimétricas na posse de cada um dos intervenientes sendo que o resto da comunicação é efetuada sobre cifra simétrica tendo por base a chave de sessão, combinando assim um Key Encapsulation Mechanism (KEM) com um Data Encryption Mechanism (DEM).

2.3.5 Sínteses criptográficas

Uma síntese criptográfica (ou *hash*) é o resultado da aplicação de uma algoritmo de síntese sobre qualquer tipo de dados de tamanho arbitrário. Uma função de síntese, H, executa uma operação sobre quaisquer dados, M, transformando-os num valor de tamanho fixo $h = H(M)$. O objetivo destas funções é garantir a integridade dos dados. Qualquer alteração num ou mais *bits* dos dados produzirá, com grande probabilidade, uma síntese distinta, sendo assim possível validar se os dados foram alterados durante a comunicação.

Uma função de síntese deve garantir quatro propriedades essenciais:

- É computacionalmente fácil gerar a síntese h para qualquer dado M.
- É computacionalmente inexequível reverter uma síntese h de forma a obter os dados que o geraram.
- É computacionalmente inexequível modificar um dado M sem que a sua síntese h não se modifique.
- É computacionalmente inexequível definir dois dados, M_1 e M_2, que originem o mesmo valor de síntese h.

Na sua essência, os algoritmos de síntese criptográfica baseiam-se na aplicação de uma função de compressão sobre os dados múltiplas vezes [33]. Esta função e o número de vezes que é aplicada varia de algoritmo para algoritmo.

2.3.6 Message Authentication Codes (MACs)

Os Message Authentication Codes (MACs) são um mecanismo utilizado para verificar a integridade de um conjunto de dados, gerando um código de autenticação de tamanho fixo a partir de uma mensagem de tamanho aleatório. Um MAC é calculado com base numa chave simétrica partilhada entre dois intervenientes, o que garante também a autenticação do emissor da mensagem, desde que a referida chave seja apenas partilhada entre duas entidades.

2.3.7 Assinaturas digitais

Uma assinatura digital (e a respetiva validação) é um par de operações de criptografia assimétrica com o objetivo de autenticar qualquer informação, bem como garantir o não-repúdio do seu envio e/ou criação por parte de um emissor, protegendo ainda a sua integridade.

O processo de assinatura e respetiva validação está retratado na Figura 3 [33]. Aqui, *Alice* calcula a síntese h de um dado conjunto de informação M e assina-o, utilizando para isso a sua chave privada, PR_{Alice}, e um algoritmo de assinatura: $Sig = S(PR_{Alice}, h)$. M é concatenado aos dados e toda esta informação é enviada. Por conseguinte, *Bob* requisita a chave pública de *Alice*, PU_{Alice}, de um repositório de chaves públicas para que, com esta

chave e um algoritmo de validação de assinatura consiga reverter o processo de assinatura, obtendo a síntese original: $Val = V(PU_{Alice}, Sig)$. De seguida, *Bob* computa a síntese de M localmente e, caso coincida com a síntese assinada digitalmente, garante não só a integridade dos dados mas também que foi realmente *Alice* quem os enviou.

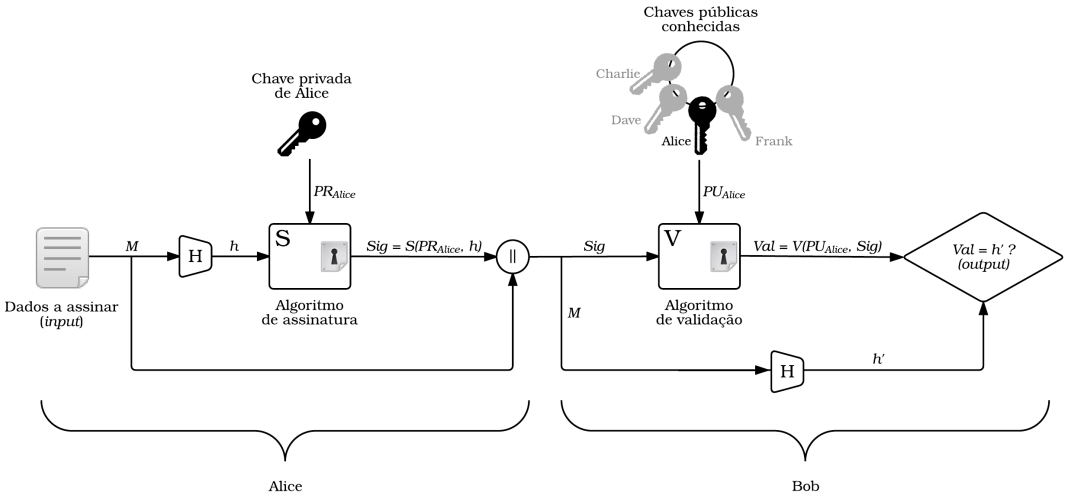

Figura 3: Modelo de assinatura/validação de assinatura digital

2.3.8 Certificados digitais

Um certificado digital é uma estrutura de dados que associa uma chave pública à identidade do seu utilizador assegurando que esta pertence, de facto, a uma entidade. Esta associação é efetuada através de uma assinatura digital sobre os dados a integrar no certificado, garantindo assim a não forgeabilidade dos dados em questão, tornando a informação indissociável da chave pública.

O tipo de certificados mais comuns está definido na norma **X.509** [7]. Este é um formato de certificado que associa a uma chave pública informações tais como o nome do seu proprietário, o nome da entidade que assinou o certificado, a sua data de expiração, utilizações que a chave pode ter, entre outros.

No que concerne a entidades que podem assinar certificados, estes podem ser assinados pelo próprio proprietário da chave pública (fazendo-o com a sua chave privada e constituíndo, assim, um certificado auto-assinado) ou, para lhe conferir um maior grau de confiabilidade, podem ser assinados por **Autoridades de Certificação**.

2.3.9 Autoridade de Certificação

Uma Autoridade de Certificação (ou, simplesmente, CA, de *Certification Authority*) é uma entidade confiável utilizada para assinar certificados digitais, conferindo-lhes um maior

grau de confiança (tal como, na vida real, um notário reconhece assinaturas). Usualmente, as CAs gerem também listas de revogação de certificados (ou CRLs, de *Certification Revocation Lists*) que contêm uma lista de certificados que deixaram de ser atestados por si (por exemplo, porque a chave privada associada à chave pública do certificado foi comprometida).

Neste contexto, de modo a validar se um dado certificado está devidamente assinado por uma CA, a validação deverá ser feita com base na chave pública da referida entidade, sendo que esta ainda pode ser assinada por uma outra CA superior, existindo assim uma cadeia de confiança entre uma CA de topo (ou CA *root*) e uma ou mais CAs subordinadas (ou Sub-CAs).

Certificados digitais podem ser facilmente encontrados na *Internet*, pois são indispensáveis para, por exemplo, implementar uma ligação HTTPS a qualquer página *Web* segura. A Figura 4 apresenta um exemplo deste tipo de certificados, onde é visível informação como o nome da entidade proprietária do mesmo, a sua data de expiração, entre outras informações. Na parte superior da imagem encontra-se a ilustração da cadeia de confiança que atesta este certificado: Existe uma CA de topo denominada "UTN-USERFirst-Hardware" que, por sua vez, assina uma CA subordinada "TERENA SSL CA". Esta última assina o certificado da entidade "www.fc.ul.pt".

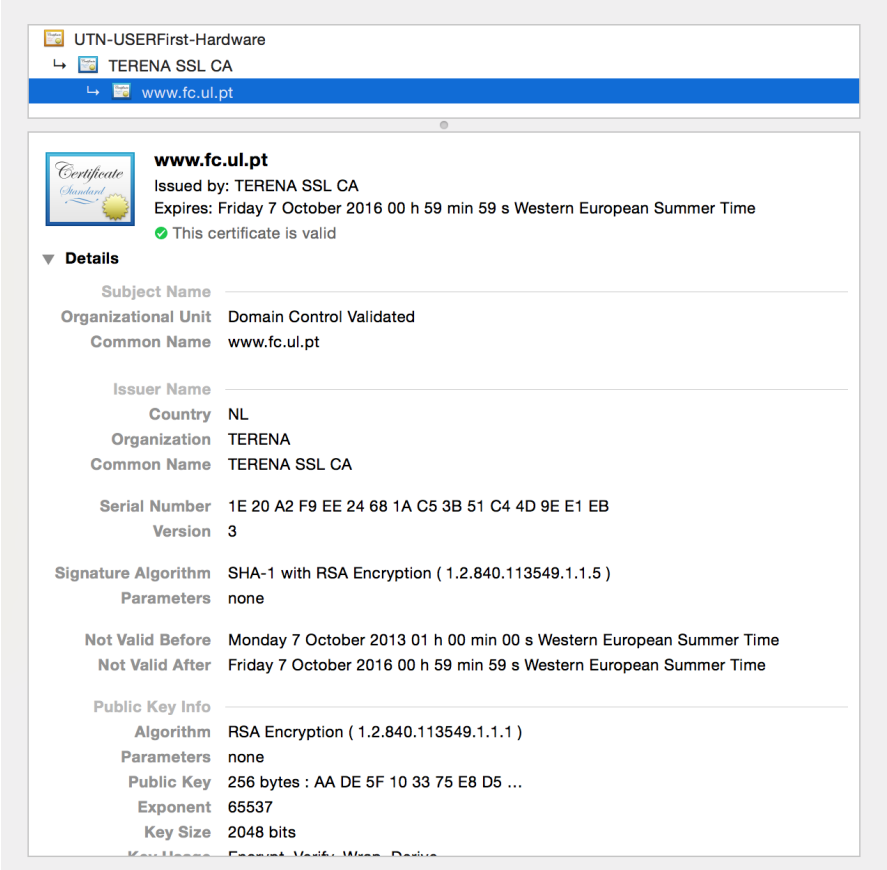

Figura 4: Exemplo de certificado digital X.509 e cadeias de certificação

2.3.10 Frescura e ordem

Em determinados protocolos de segurança é necessário garantir que as mensagens trocadas entre os intervenientes foram geradas recentemente ou que não foram trocadas de ordem por ação indevida. Para este efeito, existe o conceito de *nonce*.

O termo *nonce* deriva de *number used once*. Tal como o nome indica, é um valor variável no tempo e com baixa probabilidade de se repetir. Pode ser um valor aleatório gerado em cada utilização, uma marca temporal (*timestamp*), um número de sequência ou uma combinação entre qualquer um destes [28].

O uso destes valores pretendem cumprir diferentes objetivos:

- **Prevenção de ataques de repetição:** Um valor aleatório trocado juntamente com uma mensagem previne que esta seja reenviada ao destinatário sem que este detete este ataque.
- **Frescura:** A presença de uma marca temporal numa mensagem comprova que a informação é recente. A falta de frescura de uma mensagem pode facilmente ser detetada caso a marca temporal denote um momento no futuro ou no passado distante.
- **Ordem:** Um valor sequencial trocado numa comunicação entre os intervenientes na mesma garante a correta ordem da informação.

2.4 Algoritmos criptográficos

Nesta secção apresentam-se algoritmos criptográficos que implementam alguns dos mecanismos de segurança observados atrás.

2.4.1 Triple Data Encryption Standard (3DES)

O 3DES [27] (ou Triple Data Encryption Algorithm (TDEA)) é um algoritmo de criptografia simétrica baseado num outro algoritmo criado nos anos 70: o Data Encryption Standard (DES). No DES, os dados são cifrados em blocos de 64 *bits* utilizando uma chave secreta de 56 *bits*. O algoritmo transforma o *input* inicial num *output* (mensagem cifrada) de 64 *bits* aplicando uma série de 16 execuções de permutações e substituições aos dados utilizando em cada execução uma sub-chave derivada da chave secreta inicial. O processo de decifra faz-se executando o mesmo algoritmo de cifra dos dados mas invertendo a ordem de utilização das sub-chaves.

Com o decorrer dos anos e com o aumento da capacidade computacional, comprometer a chave de 56 *bits* do DES tornou-se exequível num curto espaço de tempo. Em 1998 foi possível, por força bruta, comprometer uma chave DES em cerca de 56 horas [13]. Por esta razão, surgiu a necessidade de aumentar a segurança do algoritmo de cifra simétrica mais utilizada nesta altura.

O 3DES veio resolver este problema sem que fosse preciso redesenhar o DES ou, até mesmo, desenvolver um novo algoritmo. Desta forma, o 3DES introduziu o conceito de **cifra múltipla**, em que o algoritmo é aplicado várias vezes. Na sua primeira execução, o algoritmo é executado sobre a mensagem em claro. De seguida, o *output* dessa mesma execução é reutilizado e passado como *input*, novamente, ao algoritmo. No caso do 3DES o algoritmo DES é aplicado três vezes com recurso a três sub-chaves de uma chave simétrica K partilhada entre os intervenientes: K_1, K_2 e K_3, sendo que estas são obtidas de forma diferente, consoante o tamanho de K.

Cifra

O esquema de cifra 3DES está representado na Figura 5 e é dividido em três passos de execução:

1. Aplicar o processo de cifra sobre a mensagem em claro M, utilizando a chave K_1.
2. Seja A o resultado da execução do ponto 1. Aplicar o processo de decifra sobre A utilizando a chave K_2.
3. Seja B o resultado da execução do ponto 2. A mensagem cifrada, C, será o resultado de aplicar o processo de cifra sobre B utilizando a chave K_3.

Transpondo o processo para uma fórmula sucinta, este pode ser definido por:

$$C = E(K_3, D(K_2, E(K_1, M)))$$

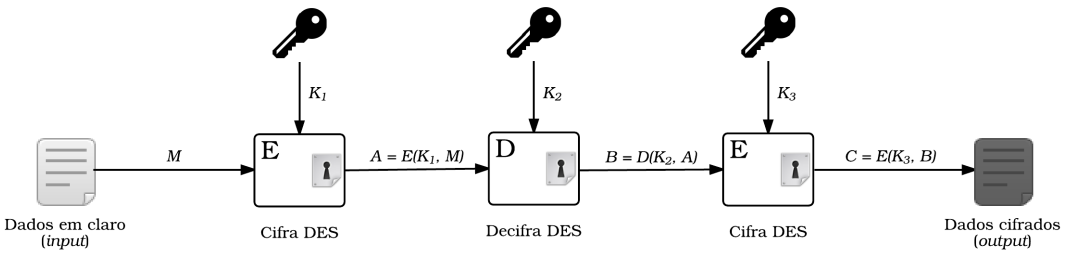

Figura 5: Algoritmo de cifra 3DES

Decifra

O esquema de decifra 3DES está representado na Figura 6. É muito similar ao processo de cifra, porém, com ligeiras diferenças:

1. Aplicar o processo de decifra sobre a mensagem cifrada C, utilizando a chave K_3.
2. Seja B o resultado da execução do ponto 1. Aplicar o processo de cifra sobre B utilizando a chave K_2.
3. Seja A o resultado da execução do ponto 2. A mensagem em claro, M, será o resultado de aplicar o processo de decifra sobre A utilizando a chave K_1.

Transpondo para uma fórmula sucinta, uma mensagem decifrada via 3DES pode ser definida por:

$$M = D(K_1, E(K_2, D(K_3, C)))$$

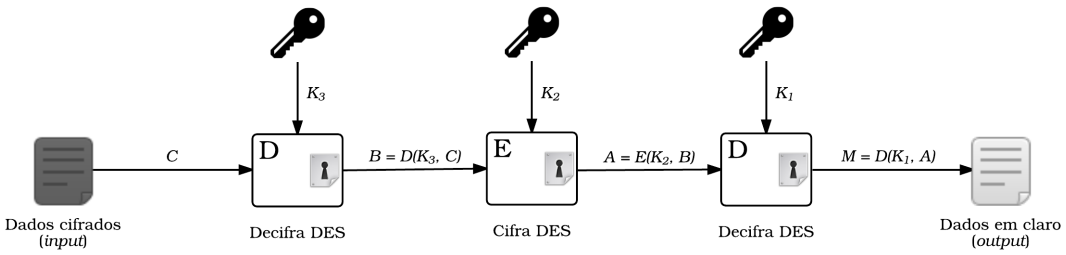

Figura 6: Algoritmo de decifra 3DES

As diferenças entre o esquema de cifra e decifra estão relacionadas com a ordem de utilização das sub-chaves e com as funções executadas em cada um dos casos: A cifra 3DES é executada invocando a função de cifra DES, seguido da decifra e novamente a cifra, com as chaves K_1, K_2 e K_3, respetivamente. A decifra 3DES executa-se aplicando a função de decifra DES, seguido da cifra e novamente a decifra, com as chaves K_3, K_2 e K_1, respetivamente.

Tamanho das chaves criptográficas

No 3DES a chave secreta K pode ter um de três tamanhos, a saber:

- 168 *bits*, ou **chaves Triple-length**: Com chaves deste tamanho, o algoritmo 3DES é executado na sua variante mais segura, pois a chave secreta K é dividida em três partes iguais de 56 *bits*. É o caso em que o algoritmo usa $K_1 \neq K_2 \neq K_3$.
- 112 *bits*, ou **chaves Double-length**: Aqui o grau de segurança é ligeiramente menor do que na variante anterior. No entanto, ainda não existem provas de que é possível comprometer uma chave deste tamanho em tempo útil. Neste caso, a chave é dividida em duas partes iguais de 56 *bits* sendo K_1 a primeira metade, K_2 a segunda metade e $K_3 = K_1$.
- 56 *bits*: Inseguro. Usar uma chave deste tamanho no 3DES tem um grau de segurança igual ao algoritmo DES, pois se $K_1 = K_2 = K_3 = K$, no processo de cifra, $C = E(K, D(K, E(K, M))) = E(K, M)$.

2.4.2 Rivest-Shamir-Adleman (RSA)

O RSA [5] foi desenvolvido em 1977 (apesar de ter sido publicado apenas no ano seguinte) por três professores do Massachusetts Institute of Technology (MIT): Ron Rivest, Adi Shamir e Len Adleman. O RSA cifra a informação em blocos, sendo que os dados em claro são convertidos em valores inteiros entre 0 e $n-1$ para um dado valor n. As funções de cifra e decifra são expressões matemáticas exponenciais. Para uma dada mensagem em claro M e o correspondente valor cifrado C, estas funções são definidas da seguinte forma:

$$C = M^e \mod n$$

$$M = C^d \mod n$$

Tanto o emissor como o recetor da mensagem deverão conhecer n, sendo que o emissor também deverá conhecer e. Por outro lado, só o recetor do texto cifrado deverá ser conhecedor de d. Ou seja, a chave pública RSA é o par $PU = (e, n)$ e a correspondente chave privada o par $PR = (d, n)$.

A segurança do RSA baseia-se na premissa de que é computacionalmente simples multiplicar dois números primos de forma a obter um terceiro número, mas computacionalmente difícil reverter essa operação de forma a descobrir esses mesmos números primos a partir do terceiro valor. A esse processo denomina-se de fatorização. Como se constatará abaixo, gerar um par de chaves RSA assenta, em parte, na multiplicação de dois números primos, p e q, mas deverá ser inviável fatorizar o valor do resultado dessa operação de forma a reverter o processo e saber-se que números primos deram origem a tal valor.

Geração do par de chaves RSA

A geração de um par de chaves RSA necessita que as seguintes propriedades se verifiquem:

- p e q são números primos.
- $n = p \cdot q$
- e é um valor inteiro tal que $1 < e < \phi(n)$ e e e $\phi(n)$ são coprimos[2], sendo ϕ a Função Totiente de Euler [20].
- $d \equiv e^{-1} \pmod{\phi(n)}$

Assim, tome-se o seguinte exemplo de geração de um par de chaves RSA:

[2]Dois números dizem-se coprimos (ou primos entre si) se o máximo divisor comum entre eles for 1.

1. Escolham-se dois números primos, $p = 73$ e $q = 151$.
2. $n = p \cdot q = 73 \times 151 = 11023$. n torna-se público.
3. $\phi(n) = \phi(p) \cdot \phi(q) = (p-1) \cdot (q-1) = 72 \times 150 = 10800$
4. Escolha-se um valor para e tal que este seja coprimo com $\phi(n) = 10800$ e menor que $\phi(n)$. Por exemplo, $e = 11$. e torna-se público.
5. Calcule-se $d \equiv e^{-1} \pmod{\phi(n)}$. É possível calcular d utilizando o Algoritmo de Euclides estendido [20]. Neste caso, $d = 5891$. d deve manter-se privado. p, q e $\phi(n)$ também se devem manter privados, pois podem ser utilizados para calcular d.

Desta forma, o par de chaves RSA está calculado: $PU = (11, 11023)$ é a chave pública e $PR = (5891, 11023)$ a respetiva chave privada.

Na prática, no RSA não são utilizados valores tão pequenos para p e q. Tipicamente estes valores são de uma ordem de grandeza bastante superior (isto é, com centenas de *bits*) e os seus comprimentos deverão ser similares entre si de maneira a inviabilizar a fatorização rápida de n em primos.

Cifra e decifra RSA

Como foi visto atrás as funções de cifra e decifra RSA são definidas, respetivamente, pelas fórmulas $C = M^e \mod n$ e $M = C^d \mod n$. A Figura 7 comprova que este algoritmo é funcional, mostrando como *Alice* deve cifrar a sequência de *bytes* de exemplo, `0x123423451ABC`, e como *Bob* a deve decifrar. O par de chaves de *Bob* é o mesmo calculado atrás: A chave pública $PU = (11, 11023)$ e a respetiva componente privada $PR = (5891, 11023)$. O diagrama divide-se em sete passos sendo que os primeiros três referem-se ao cálculo do par de chaves de *Bob* e publicação da componente pública, algo que só é necessário efetuar a primeira vez que se deseje receber informação cifrada por meio do RSA.

1. *Bob* inicia o processo de geração do seu par de chaves. Após escolher dois valores primos, p e q, calcula a sua chave pública $PU = (11, 11023)$.
2. *Bob* calcula a sua chave privada, $PR = (5891, 11023)$, mantendo-a apenas do seu conhecimento.
3. *Bob* publica a sua chave pública de forma a estar acessível para *Alice*.
4. *Alice*, que pretende enviar a sequência de *bytes* `0x123423451ABC`, começa por dividir os dados em blocos de igual tamanho[3]. No caso, por questões de simplificação, dividiram-se os dados iniciais em três blocos de 2 *bytes*, dando origem aos blocos $P_1 =$`0x1234`, $P_2 =$`0x2345` e $P_3 =$`0x1ABC`.
5. Com base na chave pública de *Bob*, *Alice* cifra a informação a enviar, bloco a bloco. São calculados os blocos cifrados $C_1 =$`0x2891`, $C_2 =$`0x2467` e $C_3 =$`0x0741`. *Alice* envia a *Bob* estes três blocos.

[3]Na prática, no RSA os dados são divididos em blocos de i *bits* tal que $2^i < n \leq 2^{i+1}$

6. Utilizando a sua chave privada, *Bob* decifra os blocos que recebe, um por um, revertendo a cifra executada com a sua chave pública.
7. Finalmente, *Bob* funde os blocos decifrados e tem assim em sua posse a sequência de *bytes* `0x123423451ABC`.

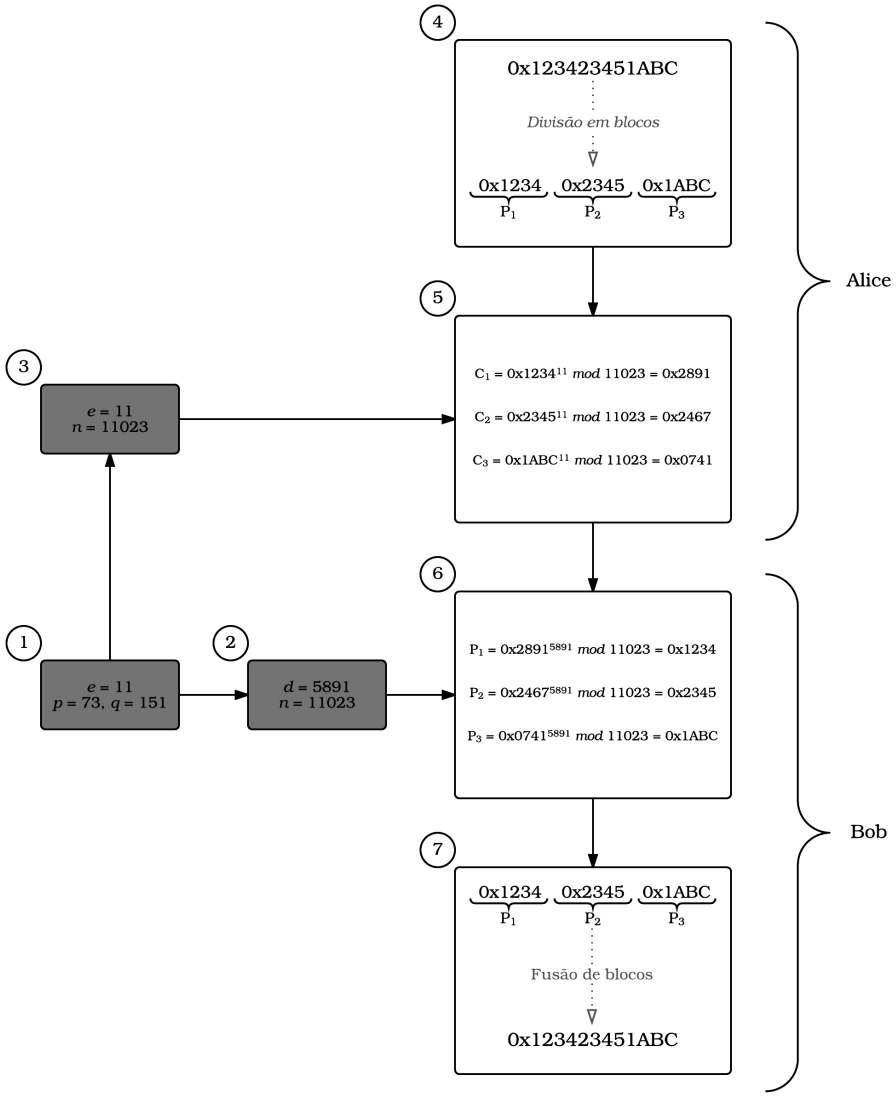

Figura 7: Diagrama de cifra e decifra RSA

Geração e validação de assinatura digital RSA

As funções de geração e validação de assinaturas RSA são muito similares às funções de cifra e decifra, respetivamente, mas utilizando a chave privada no primeiro caso e a pública no outro. Assim, os procedimentos de assinar e validar assinaturas podem definir-se da seguinte forma:

$$S = M^d \mod n$$

$$M = S^e \mod n$$

Os cálculos matemáticos envolvidos no processo são em tudo semelhantes aos que foram vistos na Figura 7 mas, desta feita, utilizando o valor de d onde atrás se utilizou e e vice-versa. Isto mostra que uma operação sobre quaisquer dados utilizando e é anulada executando a mesma operação com d (tal como na cifra/decifra de dados) mas o contrário também é válido.

2.4.3 Diffie-Hellman

O objetivo do algoritmo Diffie-Hellman [12] é acordar uma chave secreta de forma segura entre dois participantes, utilizando para tal dois valores gerados por cada participante e mantendo um deles público e o outro privado, constituíndo assim um algoritmo de criptografia assimétrica com o intuito de dois interlocutores acordarem uma chave simétrica entre si. Baseia-se na dificuldade de calcular logaritmos discretos e pode ser definido da seguinte forma:

1. *Alice* e *Bob* acordam entre si dois valores inteiros, q e α. Estes valores são de domínio público. q é um número primo e α uma raíz primitiva de q.
2. *Alice* gera um valor inteiro $X_A < q$ aleatoriamente, que deverá manter secreto, calcula um valor público, $Y_A = \alpha^{X_A} \mod q$, e envia-o a *Bob*.
3. Por conseguinte, *Bob* gera um valor inteiro $X_B < q$ aleatoriamente, que deverá manter secreto, calcula um valor público, $Y_B = \alpha^{X_B} \mod q$, e envia-o a *Alice*.

Finalmente, a chave partilhada é calculada localmente em cada uma das partes:

- *Alice* calcula: $K = (Y_B)^{X_A} \mod q$.
- *Bob* calcula: $K = (Y_A)^{X_B} \mod q$.

Matematicamente, prova-se que K é idêntico em ambas as partes pois:

$$\begin{aligned}
K &= \boldsymbol{(Y_B)^{X_A} \mod q} \\
&= (\alpha^{X_B} \mod q)^{X_A} \mod q \\
&= (\alpha^{X_B})^{X_A} \mod q \\
&= \alpha^{X_B X_A} \mod q \\
&= (\alpha^{X_A})^{X_B} \mod q \\
&= (\alpha^{X_A} \mod q)^{X_B} \mod q \\
&= \boldsymbol{(Y_A)^{X_B} \mod q}
\end{aligned}$$

Este protocolo é vulnerável a um ataque "homem-no-meio", pois não inclui mecanismos de autenticação que o previnam. *Alice*, não tendo certezas de estar realmente a

acordar uma chave secreta com *Bob* pode, na verdade, estar a acordar a chave com um atacante, *Eve*, sendo isto também válido para *Bob*. Neste caso, quando *Alice* enviasse uma mensagem para *Bob* estaria, na verdade, a enviá-la para *Eve* que a decifraria e reencaminharia a *Bob*, cifrada agora pela chave que estes dois partilham.

De modo a não ser vulnerável a este tipo de ataques, os participantes deverão assinar digitalmente os valores trocados entre si, garantido assim a autenticidade e não-repúdio dos mesmos.

2.4.4 Algoritmos de síntese criptográfica

Alguns dos algoritmos geradores de sínteses criptográficas mais comuns são o MD5 [30], SHA-1 e SHA-2 [26]. Estes geram sínteses de tamanho 128 *bits* e 160 *bits* no caso do MD5 e SHA-1, respetivamente. O SHA-2 é, na verdade, um conjunto de seis algoritmos que produzem sínteses de 224, 256, 384 e 512 *bits*, sendo que as seis funções denominam-se SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 e SHA-512/256 (estas duas últimas são versões truncadas da variante de 512 *bits*, com ligeiras diferenças adicionais entre si). A título de exemplo, abaixo apresentam-se os valores que algumas destas funções retornam sobre os exemplos `abcd` e `Abcd`, mostrando que mesmo uma diferença mínima entre eles gera uma síntese criptográfica distinta.

MD5(`abcd`) = `e2fc714c4727ee9395f324cd2e7f331f`
MD5(`Abcd`) = `30f64f3171b1fa24a1698bdf0b435b19`

SHA-1(`abcd`) = `81fe8bfe87576c3ecb22426f8e57847382917acf`
SHA-1(`Abcd`) = `c6c42be007971599963573d1ef39aa5f6939b5dd`

SHA-256(`abcd`) = `88d4266fd4e6338d13b845fcf289579d209c897823b9217da3e161936f031589`
SHA-256(`Abcd`) = `eafd87e0761f8510a40e45cb1273de52196650710a34f80f216c361f4b6aa635`

2.4.5 Algoritmos de MAC

De modo a produzirem um código de autenticação para uma mensagem, os algoritmos de MAC recebem como *input*, para além da referida mensagem, uma chave secreta partilhada entre os dois intervenientes da comunicação. São, portanto, algoritmos de criptografia simétrica e dividem-se em dois grupos: Podem ser algoritmos baseados em sínteses criptográficas ou cifras simétricas.

O algoritmo HMAC [6] é o mais utilizado no âmbito dos algoritmos de MAC baseados em sínteses criptográficas. Este pode fazer uso de qualquer algoritmo de síntese criptográfica existente e pode ser resumindo na seguinte fórmula:

$$HMAC(K, M) = H((K \oplus opad) \parallel H((K \oplus ipad) \parallel M))$$

Em que:

- K é a chave simétrica partilhada.
- M é a mensagem a autenticar.
- H é o algoritmo de sínteses criptográficas utilizado.
- $opad$ e $ipad$ são valores constantes definidos na especificação do algoritmo.

O tamanho do *output* depende diretamente do algoritmo de sínteses criptográficas utilizado.

No campo dos algoritmos de MAC baseados em cifras simétricas, o algoritmo CMAC é um dos mais populares (sendo uma variante do algoritmo CBC-MAC mas corrigindo algumas das deficiências provadas do mesmo [20]). O CMAC pode fazer uso de múltiplos algoritmos de cifra simétrica, como é o caso do 3DES.

Para calcular um MAC, o CMAC inicia o processo dividindo uma mensagem M em blocos (M_1, M_2, M_3, ... , M_n). De seguida, o algoritmo procede do seguinte modo [25]:

$$\begin{aligned} C_1 &= E(K, M_1) \\ C_2 &= E(K, (M_2 \oplus C_1)) \\ C_3 &= E(K, (M_3 \oplus C_2)) \\ &\dots \\ C_n &= E(K, (M_n \oplus C_{n-1} \oplus K')) \end{aligned}$$

Em que:

- E é o algoritmo de cifra utilizado.
- K é chave simétrica partilhada.
- K' é uma sub-chave derivada de K de acordo com a especificação do algoritmo.
- M_1, M_2, M_3, ... , M_n é a mensagem a autenticar dividida em n blocos.
- C_1, C_2, C_3, ... , C_n é o resultado da cifra dos respetivos blocos que compõem a mensagem a autenticar.

Finalmente, o valor do MAC são os *bits* mais significativos de C_n, sendo que o número de *bits* não está definido, podendo ser escolhido consoante se deseje.

A validação dos MACs é transversal a qualquer tipo de algoritmo utilizado. Dado que o emissor envia a mensagem juntamente com o respetivo MAC, e admitindo que o recetor conhece o algoritmo que gerou o código de autenticação, este só necessita de gerar o MAC localmente com a mesma chave K e verificar se coincide com o que recebeu do emissor.

2.5 *Scyther*: Analisador de protocolos de segurança

O *Scyther* [10] é uma ferramenta de análise formal de protocolos de segurança que assume o modelo de "criptografia perfeita", no qual se admite que todas as funções criptográficas são perfeitas, isto é, nenhum adversário é capaz de obter informação a partir de dados cifrados exceto se tiver conhecimento da chave que os decifre.

Com base nesta premissa, é possível utilizar esta ferramenta com o intuito de encontrar vulnerabilidades de segurança na construção de um protocolo. Para isso, o *Scyther* recebe como *input* uma descrição do protocolo (escrita numa linguagem específica, denominada Security Protocol Description Language (SPDL)) que inclui a especificação das propriedades de segurança que se pretendem atingir, sob a forma de *security claims*. A sintaxe destas e de outras propriedades são descritas de seguida. A compreensão desta notação SPDL é fundamental para o trabalho pois o *Scyther* será utilizado para análise de protocolos mais adiante.

2.5.1 Sintaxe do *Scyther*

De forma a escrever e entender a sintaxe SPDL são necessários os seguintes conceitos [11]:

Protocolo

Em SPDL, um protocolo é definido por um nome e um conjunto de intervenientes (*roles*). No caso, apresenta-se o protocolo $MyProt$ que é posto em prática entre dois *roles*: I e R.

```
1  protocol MyProt(I, R){
2      role I {
3          ...
4      }
5
6      role R {
7          ...
8      }
9  }
```

Sintaxe SPDL 2.1: Definição de protocolo

Evento

Na notação SPDL, os eventos podem ser de três tipos: $send$, $recv$ e $claim$. Opcionalmente, pode acrescentar-se um sufixo numérico para auxiliar na leitura dos fluxos do protocolo.

Os primeiros dois servem para denotar o envio e receção de dados, respetivamente. Uma comunicação entre os *roles* I e R é definida como:

```
1  send_1(I, R, x);
2  recv_2(R, I, y);
```

Sintaxe SPDL 2.2: Definição de eventos $send$ e $recv$

O evento $claim$ é utilizado na modelação das propriedades de segurança que se querem garantir no protocolo, sendo estas as características que o *Scyther* vai testar. Por exemplo, o evento $claim$ abaixo garante que a variável x mantém-se secreta no decorrer do protocolo, no contexto do *role* I:

```
1  claim_1(I,Secret,x);
```

Sintaxe SPDL 2.3: Definição de evento $claim$

A lista completa de *claims* existentes na linguagem SPDL é apresentada abaixo e divide os *claims* existentes como podendo garantir **confidencialidade**, **autenticação**, **integridade** ou **ordem**:

- Confidencialidade
  - $Secret$ [11]: Utilizado para expressar que um dado termo não poderá ser revelado a um adversário, mesmo que este atravesse redes potencialmente inseguras. Por exemplo: $claim(R, Secret, x)$ denota que o termo x, utilizado no contexto do *role* R, se manterá secreto no decorrer do protocolo.

  - SKR [11]: Semelhante ao anterior. Utilizado para garantir especificamente a confidencialidade de uma chave de sessão trocada entre os interlocutores. Por exemplo, $claim(R, SKR, K)$ significa que uma chave de sessão K utilizada pelo *role* R se manterá secreta no decorrer do protocolo.
- Autenticação
  - $Alive$ [19]: Utilizado quando se pretende garantir um nível mínimo de autenticidade. Se um *role* I executa um protocolo com o *role* R, então garante-se que é R quem realmente está a responder, ou seja, assegura-se a *aliveness* do *role* R. Este requisito define-se por $claim(R, Alive)$ e utiliza-se quando se deseja garantir uma **autenticação unilateral**.

  - $Weakagree$ [19]: Esta propriedade reforça um pouco mais a anterior. Se um *role* I executa um protocolo com o *role* R, então garante-se a R que está a comunicar com I, ou seja, assegura-se o *weak agreement* do *role* R e define-se por $claim(R, Weakagree)$ e utiliza-se quando se deseja garantir uma **autenticação mútua**.

- Integridade e ordem
  - $Niagree$ [9]: Se dois *roles* I e R estão mutuamente autenticados e a trocar mensagens entre si, de acordo com a especificação do protocolo, então pode-se garantir a **integridade** das mensagens. Define-se por $claim(R, Niagree)$, no contexto do *role* R. O termo $Niagree$ deriva de *non-injective agreement*.

  - $Nisynch$ [9]: Esta propriedade garante um nível de autenticação superior. Um *role* R respeita esta condição se:
    - Os conteúdos dos eventos $recv$ são idênticos aos dos correspondentes eventos $send$ (isto é, preserva-se a **integridade** das mensagens).
    - A **ordem** dos eventos respeita a especificação do protocolo.

    $claim(R, Nisynch)$ assegura que um *role* R garante estas propriedades. O termo $Nisynch$ deriva de *non-injective synchronization*.

  - $Commit, Running$ [31]: Utilizado para garantir a **integridade** de um valor específico durante o protocolo, e não da troca de mensagens como um todo (tal como no caso do *claim* $Niagree$). Útil, por exemplo, para afirmar que os *roles* validaram corretamente um *nonce* trocado entre si. Por exemplo, um *role* I que envia um *nonce*, n, a R, garante que terminou a execução do protocolo com R e com o valor de n acordado entre as partes colocando, no final da especificação do *role*, o *claim* $claim(I, Commit, R, n)$. Por outro lado R, para garantir que está realmente a executar o protocolo com I e acordam o valor de n, define $claim(R, Running, I, n)$ após ter recebido n.

Role

Um *role* define um interveniente no protocolo, cuja especificação contém uma sequência de eventos, que podem ser do tipo $send$, $recv$ ou $claim$ definidos anteriormente.

```
 1  role I {
 2      fresh x: Nonce;
 3      var y: Nonce;
 4
 5      send_1(I,R,x);
 6      recv_2(R,I,y);
 7  }
 8
 9  role R {
10      var x: Nonce;
11      fresh y: Nonce;
12
13      recv_1(I,R,x);
14      send_2(R,I,y);
15  }
```

Sintaxe SPDL 2.4: Definição de *Roles*

Neste exemplo é possível constatar a troca de dois *nonces* - x e y - entre os *roles* I e R. De notar que a palavra reservada *var* serve para declarar uma variável que guardará um qualquer conteúdo. A palavra reservada *fresh* denota um valor gerado no momento.

Elementos criptográficos

Na notação SPDL é também possível descrever operações criptográficas simétricas ou assimétricas, bem como a utilização de sínteses.

Por exemplo, a ação de enviar x cifrado pela chave simétrica K por parte do *role* I a R define-se por:

```
1  send_1(I,R,{x}K);
```

Sintaxe SPDL 2.5: Definição de ação com chave simétrica

A notação da componente de criptografia assimétrica é semelhante. Caso o *role* I necessite de enviar x a R cifrado pela sua chave pública, esta ação define-se por:

```
1  send_1(I,R,{x}pk{R});
```

Sintaxe SPDL 2.6: Definição de ação com chave pública

A operação análoga mas com recurso à chave privada de I define-se por:

```
1  send_1(I,R,{x}sk{I});
```

Sintaxe SPDL 2.7: Definição de ação com chave privada

É também possível utilizar o conceito de sínteses criptográficas. Para isso, define-se abstratamente uma função de síntese que pode ser utilizada onde for necessário. Por exemplo, caso o *role* I queira enviar a R a síntese de x, isto seria descrito por:

```
1  hashfunction Hash;
2  send_1(I,R,Hash(x));
```

Sintaxe SPDL 2.8: Definição de algoritmo de sínteses criptográficas

2.6 Conclusão do capítulo

Este capítulo apresentou vários tópicos relacionados com o trabalho desenvolvido e com segurança da informação em geral, que serão úteis para definir o protocolo de injeção remota de chaves em POSs. Para melhor compreender o que são estes equipamentos

e quais os mecanismos de injeção de chaves atualmente praticados em Portugal, bem como uma possível forma de evoluir estes métodos, o próximo capítulo faz o devido enquadramento.

Capítulo 3

Enquadramento técnico

Este capítulo apresenta sucintamente o que são Points of Sale (POSs) e quais as suas principais características. Define as chaves criptográficas de que estes terminais necessitam e os métodos existentes de carregamento deste material criptográfico, analisando-os e apontando as suas vantagens e desvantagens. Apresenta, também, o conceito de injeção remota de chaves como uma possível forma de evoluir os atuais protocolos de inicialização destes terminais.

3.1 Points of Sale (POSs)

Os Points of Sale (POSs) (também apelidados de Terminais de Pagamento Automático (TPAs)) são equipamentos que permitem o pagamento de bens com cartões bancários. A Figura 8 apresenta alguns POSs com diferentes formatos[1].

Figura 8: Exemplos de vários tipos de POSs

[1]Fonte: `http://www.ingenico.com/en/products/payment-terminals/`

No canto superior esquerdo da imagem está um exemplo do formato mais comum dos POSs, com um visor simples e um teclado. Contudo, estes equipamentos podem ser mais complexos. Por exemplo, o visor pode ser mais interativo e sensível ao toque para permitir ao seu utilizador providenciar a sua assinatura (canto superior direito) ou serem construídos de forma a estarem encastrados (parte inferior da imagem).

3.1.1 Principais características

Internamente estes equipamentos são muito semelhantes entre si, independentemente do fabricante ou formato. Os componentes mais comuns no seu interior são:

- Microprocessador para executar as operações que um POS deve garantir.
- Memória RAM, tipicamente cifrada.
- Trusted Platform Module (TPM), para guardar de forma segura as chaves criptográficas do equipamento. Por vezes também chamado de Security Application Module (SAM).
- Leitor de *chip*.
- Leitor de banda magnética.
- Sensores que detetam violação (*tampering*) do equipamento.

Todos estes equipamentos devem ser especialmente concebidos para garantir a integridade e confidencialidade de dados sensíveis (como o número do cartão utilizado, o PIN, etc.) e também das chaves criptográficas utilizadas numa transação, de maneira a serem imúnes aos seguintes vetores de ataque [16]:

- **Penetração:** Perfuração e/ou abertura indevida de um dispositivo criptográfico para obter informação sensível nele guardada (por exemplo, chaves criptográficas).
- **Monitorização:** Monitorizar radiações eletromagnéticas do dispositivo com o objetivo de aceder a informação sensível contida no equipamento; Ou intercetar visualmente, auricularmente ou eletronicamente dados secretos a serem injetados no dispositivo, por exemplo, no momento da sua configuração inicial.
- **Manipulação:** Envio não autorizado de *inputs* ao dispositivo para que seja causado comprometimento de dados sensíveis, por exemplo, fazendo-o entrar no modo de *debug* onde pode ser possível reconfigurar o equipamento maliciosamente.
- **Modificação:** Alteração não autorizada das características físicas ou lógicas do dispositivo criptográfico, como por exemplo a inserção de um componente malicioso que comprometa o PIN inserido, capturando-o entre o momento em que é digitado pelo cliente e antes de ser cifrado e enviado para o processador de transações, ou a alteração das chaves criptográficas.
- **Substituição:** Substituição indevida de um equipamento criptográfico por outro, potencialmente malicioso, que comprometa toda a informação que por ele passe.

Para mitigar qualquer um destes vetores de ataque, os POSs são desenvolvidos tendo em vista três características fundamentais. Estes equipamentos criptográficos devem ser:

- ***Tamper Evident:*** Após um ataque sobre um dispositivo criptográfico este mostra evidências de que foi alvo de uma tentativa de comprometimento do mesmo, seja com ou sem sucesso. Estas evidências podem passar por simples marcas danosas no POS.
- ***Tamper Resistant:*** O equipamento tem capacidade de bloquear ataques que visem comprometer a confidencialidade ou integridade de informação sensível que passe pelo seu interior.
- ***Tamper Responsive:*** O dispositivo possui auto-defesas que pode utilizar caso seja alvo de ataque. Por exemplo, em caso de tentativa de comprometimento das chaves criptográficas um POS deve ter mecanismos de eliminação imediata das mesmas.

3.1.2 Chaves criptográficas

O transporte seguro da informação sensível respeitante a uma transação desde o POS até ao processador da mesma baseia-se em quatro chaves simétricas 3DES de 112 *bits* (*Double-length*) de longa duração:

- **1 Terminal Master Key (TMK):** Chave "mestra" do POS, utilizada para cifrar as restantes três chaves.
- **3 Terminal Working Keys (TWKs):**
  - Chave de MAC: Necessária para a geração e validação de MACs que são enviados juntamente com as mensagens trocadas com o processador da transação, de forma a ambos validarem a integridade e autenticidade dos dados.
  - Chave de PIN: Utilizada para cifrar o PIN digitado pelo cliente antes de ser enviado ao processador da transação para sua validação.
  - Chave de cifra: Para a cifra de outros dados genéricos trocados entre o POS e o processador da transação.

Estas quatro chaves garantem a segurança do sistema de transações, pelo que o seu comprometimento pode levar a que os dados sensíveis do pagamento e do cartão utilizado sejam capturados por um atacante. Os ataques podem passar por:

- Utilizar a TMK para decifrar as restantes TWKs.
- Capturar o tráfego oriundo do POS e utilizar a chave de PIN para decifrar o PIN do cartão.
- Pelo mesmo método de captura, alterar o conteúdo dos dados a enviar para o processador de transações e recalcular o MAC da nova mensagem utilizando a respetiva chave de forma a que esta quebra de integridade não seja detetada.

- Decifrar outros dados críticos a enviar ao processador cifrados pela chave de cifra genérica (por exemplo, o número do cartão e a sua data de expiração).

Sendo que este material criptográfico dentro do POS reside no seu TPM, o comprometimento das chaves por parte de um atacante é uma tarefa complexa, dada a natureza destes módulos de segurança. Assim, o momento do carregamento destas chaves no POS revela-se a altura ideal para um atacante ficar em posse destes segredos. A secção seguinte apresenta os métodos utilizados na rede nacional de POSs para esta injeção inicial de chaves criptográficas.

3.2 Métodos existentes de injeção de chaves

Hoje em dia, em Portugal, são utilizados dois métodos de carregamento de chaves criptográficas nestes equipamentos. Ambos os procedimentos são detalhados de seguida.

3.2.1 Carregamento local centralizado

Este tipo de carregamento de chaves acarreta o peso logístico do transporte dos POSs ao centro de inicialização de terminais do processador de transações para a injeção do material criptográficos nos equipamentos, centralizando em si todo o processo. Neste método, o procedimento é o seguinte:

1. O fabricante entrega ao processador os terminais a inicializar.
2. Numa sala segura, é carregada a TMK e as TWKs em cada equipamento entregue, ligando os POSs a um Hardware Security Module (HSM) para este injetar as chaves simétricas no terminal.
3. O fabricante recolhe os POS já inicializados.

3.2.2 Carregamento local descentralizado

Este método de injeção de chaves descentraliza o carregamento de chaves, passando essa responsabilidade para os fabricantes. É o procedimento mais utilizado atualmente e executa-se do seguinte modo:

1. O fabricante e o processador estabelecem entre si uma chave simétrica de longa duração para transporte de outras chaves.
2. O fabricante solicita ao processador a geração de um determinado número de TMKs e TWKs para estes procederem à inicialização dos seus próprios terminais.
3. É criado um ficheiro XML com o material criptográfico gerado, que será disponibilizado ao fabricante através de suporte CD-ROM. As chaves criptográficas são gravadas cifradas por uma chave 3DES única por cada ficheiro XML, sendo que

esta também é enviada ao fabricante cifrada pela chave de transporte acordada anteriormente.
4. O fabricante, tendo em sua posse as chaves criptográficas, insere-as no seu Key Loading Device (KLD)[2] (que deverá estar numa sala segura).
5. Os POSs podem ser ligados ao KLD para carregamento das suas chaves criptográficas.

3.3 Análise ao cenário de injeção de chaves atual

Analisando qualquer um dos métodos estão patentes algumas desvantagens em ambos. Por um lado o primeiro método é logisticamente complexo pois obriga a que o fabricante se desloque até à sala segura do processador e aí, os responsáveis pela inicialização de terminais necessitam de desempacotar cada POS, ligá-lo ao seu HSM, proceder ao carregamento das chaves, e voltar a empacotá-los para que sejam devolvidos ao fabricante.

Por outro lado, o segundo método vem aliviar este esforço logístico, transferindo a responsabilidade da inicialização dos terminais para o respectivo fabricante. No entanto, obriga a que os fabricantes de POSs tenham de construir, nas suas instalações, salas seguras para inicialização destes terminais. Esta multiplicação de salas seguras aumenta também a superfície de ataque por vários pontos geográficos, sobretudo se estas salas não tiverem em linha de conta certos requisitos de segurança, idênticos aos do centro de inicialização de terminais do processador (por exemplo: restringir a entrada na sala a pessoal autorizado, registar entradas e saídas na sala, ou ser apenas possível aceder ao HSM/KLD mediante autenticação de duas pessoas (*dual control*)).

Finalmente, sendo estes métodos de injeção de chaves criptográficas de cariz manual e necessitando sempre de intervenção humana, nenhum deles oferece formas de atualização periódica destas chaves ao longo do ciclo de vida do POS.

Por estas razões, torna-se necessário evoluir esta metodologia de injeção manual de chaves de modo a resolver problemas como:

- Demasiada carga logística.
- Demasiada intervenção humana, suscetível a erros.
- Processo lento, sobretudo quando é feita a inicialização em massa a muitos terminais.
- Falta de processo de atualização periódica de chaves criptográficas.

Neste sentido, surgiu nos últimos anos um conceito de **injeção remota de chaves criptográficas** que vem ao encontro desta problemática e é introduzido na secção seguinte.

[2]Equipamento que, após receber um conjunto de chaves criptográficas, automatiza o processo de injeção das chaves num POS, diminuindo assim a carga de intervenção humana na inicialização do terminal.

3.4 Injeção remota de chaves

Injeção remota é o processo de distribuição de chaves simétricas de um ponto central até um equipamento final (neste contexto, um POS). Deste modo, o terminal pode ser instalado no cliente final sem ser alvo de todo o processo de injeção manual de chaves por qualquer um dos métodos apresentados anteriormente, já que este protocolo permitirá ao POS receber as chaves de forma segura desde o ponto de administração de chaves criptográficas, já no local de instalação final, simplificando o processo de logísitica entre fabricantes de POSs e o processador. Adicionalmente, este método permite substituir as chaves criptográficas num POS em caso de comprometimento das mesmas ou caso se deseje efetuar uma troca periódica deste material criptográfico para uma maior segurança.

No entanto, o desenho deste protocolo deve ter em consideração que as chaves simétricas do POS irão viajar desde um ponto central de distribuição até ele por redes potencialmente inseguras, pelo que é primordial desenvolver um método seguro de forma a não comprometer as chaves partilhadas.

Da investigação feita na parte inicial deste trabalho foram selecionados dois documentos que identificam o que deve refletir um protocolo que execute esta inicialização remota de terminais com material criptográfico simétrico:

- A norma **ANS X9.24-2** [2], que apresenta um conjunto de regras e boas práticas que devem ser cumpridas caso uma entidade financeira deseje implementar um protocolo de gestão remota de chaves simétricas.
- O documento **ASC X9 TR34** [4] que propõe uma implementação de um protocolo deste âmbito refletindo as regras do anterior.

Ambos os documentos foram publicados pela Accredited Standards Committee (ASC) (grupo acreditado pela American National Standards Institute (ANSI)), e são analisados de seguida.

3.4.1 A norma ANS X9.24-2

A norma **ANS X9.24-2** define um conjunto de requisitos e diretrizes a aplicar na gestão de chaves criptográficas no contexto de equipamentos que efetuam transacções financeiras.

Para que duas entidades partilhem uma ou mais chaves simétricas esta norma propõe três métodos de distribuição de chaves simétricas, podendo ser estabelecidas entre entidades por **transporte** ou por **acordo**. Neste contexto, as entidades envolvidas são um POS e o processador de transações.

Neste documento, a técnica sugerida de injetar chaves simétricas remotamente baseia-se em criptografia assimétrica, cujas formas de transporte são apresentadas de seguida.

Protocolo de transporte unilateral de chaves

A Figura 9 apresenta, em alto nível, como se deve executar um transporte unilateral de uma chave simétrica com uma das partes a gerar a mesma, enviando-a de seguida para o outro interlocutor. Este método divide-se em três passos:

1. O processador envia ao POS o seu certificado digital, para que este o valide.
2. O POS envia ao processador o seu certificado digital, para que este o valide.
3. O processador gera a chave simétrica a enviar, cifra-a com a chave pública do POS e assina os dados cifrados, enviando-os ao POS.

Finalmente, caso o POS valide com sucesso a assinatura dos dados, pode decifrar a chave simétrica, guardando-a no seu TPM.

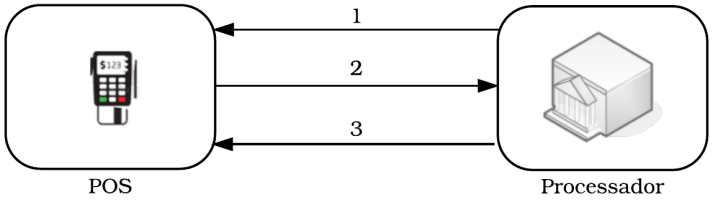

Figura 9: Esquema de protocolo de transporte unilateral de chaves

Protocolo de transporte bilateral de chaves

A Figura 10 apresenta o procedimento do transporte bilateral de chaves onde ambos os interlocutores geram informação e trocam-na entre si para, localmente, conjugarem os dados de maneira a formarem uma chave simétrica. Sucintamente, os passos são:

1. O POS envia ao processador o seu certificado digital, para que este o valide.
2. O processador envia ao POS o seu certificado digital, para que este o valide.
3. O POS gera uma chave simétrica, cifra-a com a chave pública do processador e assina os dados cifrados, enviando-os estes dados ao processador.
4. Da mesma forma, o processador gera uma chave simétrica, cifra-a com a chave pública do POS e assina os dados cifrados, enviando-os ao POS.

Caso ambos os intervenientes validem as assinaturas corretamente, cada um deles tem em sua posse duas chaves simétricas. Estas podem ser conjugadas entre si de maneira a formarem uma só chave simétrica por intermédio de um algoritmo desenhado para o efeito que as combine de forma a que não seja possível deduzir as chaves trocadas e que alimentam o referido algoritmo [17].

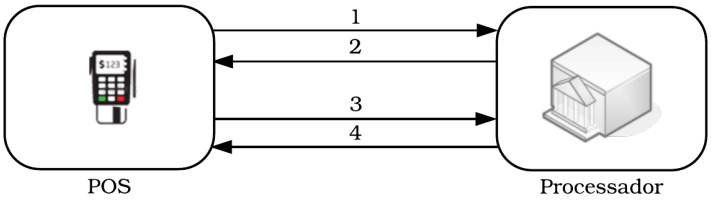

Figura 10: Esquema de protocolo de transporte bilateral de chaves

Protocolo de acordo de chaves

Neste método, ao contrário dos anteriores, a chave simétrica não é transportada de um interveniente até outro. A chave simétrica é acordada entre ambos os interlocutores. Para este estabelecimento de chaves, a norma propõe o algoritmo Diffie-Hellman como sendo seguro, desde que a informação trocada entre os intervenientes na execução deste algoritmo seja assinada digitalmente.

3.4.2 Proposta de implementação: ASC X9 TR34

Baseado na norma ANS X9.24-2, foi publicado um outro documento que apresenta uma possível implementação de um protocolo de acordo com a norma denominado **ASC X9 TR34**. O documento descreve o método de injeção remota de chaves para a situação particular em que um ponto central de gestão de chaves simétricas (o *Key Distribution Host (KDH)*) é o distribuidor de chaves para múltiplos POSs (ou *Key Receiving Devices (KRDs)*).

Protocolo

O algoritmo proposto divide-se em três fases distintas:

- **Vinculação:** Onde o KDH e o KRD trocam os seus certificados digitais entre si.
- **Transporte:** Onde o KDH envia a chave simétrica a injetar no TPM do KRD.
- **Confirmação:** Onde o KRD envia o *Key Check Value (KCV)*[3] da chave recebida ao KDH para que este confirme que a chave foi corretamente recebida.

Fase 1: Vinculação

Nesta fase, tanto o KRD como o KDH trocam entre si os seus certificados digitais, cujas chaves públicas serão utilizadas na fase seguinte. Esta vinculação desenrola-se nos seguintes passos:

[3] O KCV tem o propósito de detetar erros na transmissão de uma chave simétrica, tal como um *checksum*.

1. O KRD envia ao KDH o seu certificado digital, $Cert\{PU_{KRD}\}$.
2. O KDH recebe o certificado, efetua a sua validação e guarda-o de forma a ser utilizado na fase seguinte.
3. O KDH envia ao KRD o seu certificado digital $Cert\{PU_{KDH}\}$.
4. O KRD recebe o certificado, efetua a sua validação e guarda-o de forma a ser utilizado na fase seguinte.

A Figura 11 esquematiza esta fase de vinculação.

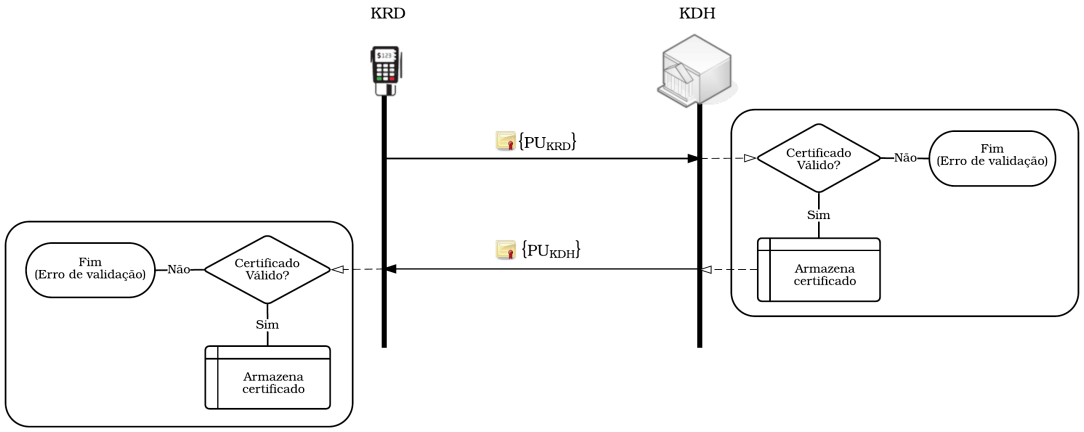

Figura 11: Protocolo TR34: Fase de vinculação

Fase 2: Transporte

Após a troca de certificados entre o KRD e o KDH, este último pode proceder à geração e envio da chave simétrica a injetar remotamente no equipamento. Esta chave simétrica será enviada numa estrutura de dados definida no documento como *key block* TR34.

O objetivo de um *key block* é associar a uma chave simétrica um conjunto de atributos que, entre outros, define o único propósito para o qual a chave pode ser utilizada. Estes atributos formam o Key Block Header (KBH). No caso do *key block* TR34, estes atributos (bem como a chave simétrica) são assinados digitalmente via RSA.

A primeira versão deste *key block* define que o formato do seu *header* é baseado no *header* de um outro *key block* - O TR31 [1] - que utiliza um MAC para proteger os atributos e a chave simétrica, em vez da assinatura digital atrás mencionada.

[4]Este campo tem o objetivo de numerar a versão do formato em utilização para facilmente ser detetada a estrutura do *key block*. Atualmente existe apenas a versão 1.

Esquematicamente, o *key block* TR34 tem o aspecto da Figura 12:

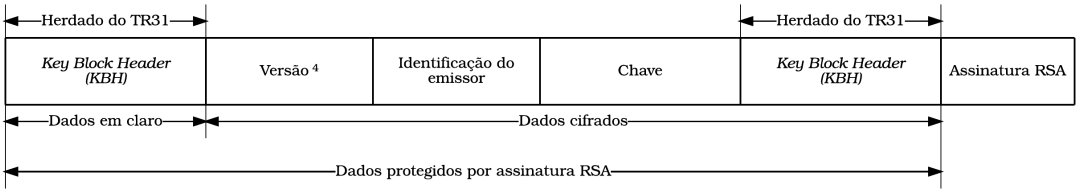

Figura 12: *Key Block* TR34

Assim, o TR34 define o seguinte procedimento para o transporte da chave simétrica:

1. O KRD gera um *nonce*, R, e envia-o ao KDH. Adicionalmente, guarda-o localmente para ser utilizado posteriormente.
2. O KDH gera a chave simétrica a injetar, K_n, guardando-a também na sua base de dados.
3. O KDH gera uma chave simétrica de sessão, K_E.
4. O KDH formata a componente cifrada do *key block* TR34, BE, utilizando K_E:

$$BE = E(K_E, (Version \parallel ID_{KDH} \parallel K_n \parallel KBH))$$

5. O KDH cifra K_E com a chave pública do KRD:

$$EncryptedKey = E(PU_{KRD}, K_E)$$

6. O KDH envia ao KRD o *nonce* R, $EncryptedKey$ e BE, juntamente com a assinatura destes dados e uma lista de revogação de certificados:

$$R \parallel KBH \parallel EncryptedKey \parallel BE \parallel$$
$$S(PR_{KDH}, (R \parallel KBH \parallel EncryptedKey \parallel BE)) \parallel CRL_{CA\_KDH}$$

7. O KRD recebe estes dados e:
   (a) Verifica se $Cert\{PU_{KDH}\}$ não se encontra na lista CRL_{CA_KDH}.
   (b) Valida a assinatura dos dados recebidos utilizando a chave pública de KDH.
   (c) Confirma se o *nonce* R recebido é o mesmo que enviou no início desta fase.
   (d) Decifra K_E; Decifra BE utilizando K_E e verifica se:
      i. ID_{KDH} coincide com a entidade proprietária do certificado $Cert\{PU_{KDH}\}$.
      ii. o valor de KBH em claro coincide com o valor de KBH cifrado (dentro de BE).

A Figura 13 esquematiza esta fase de transporte da chave simétrica.

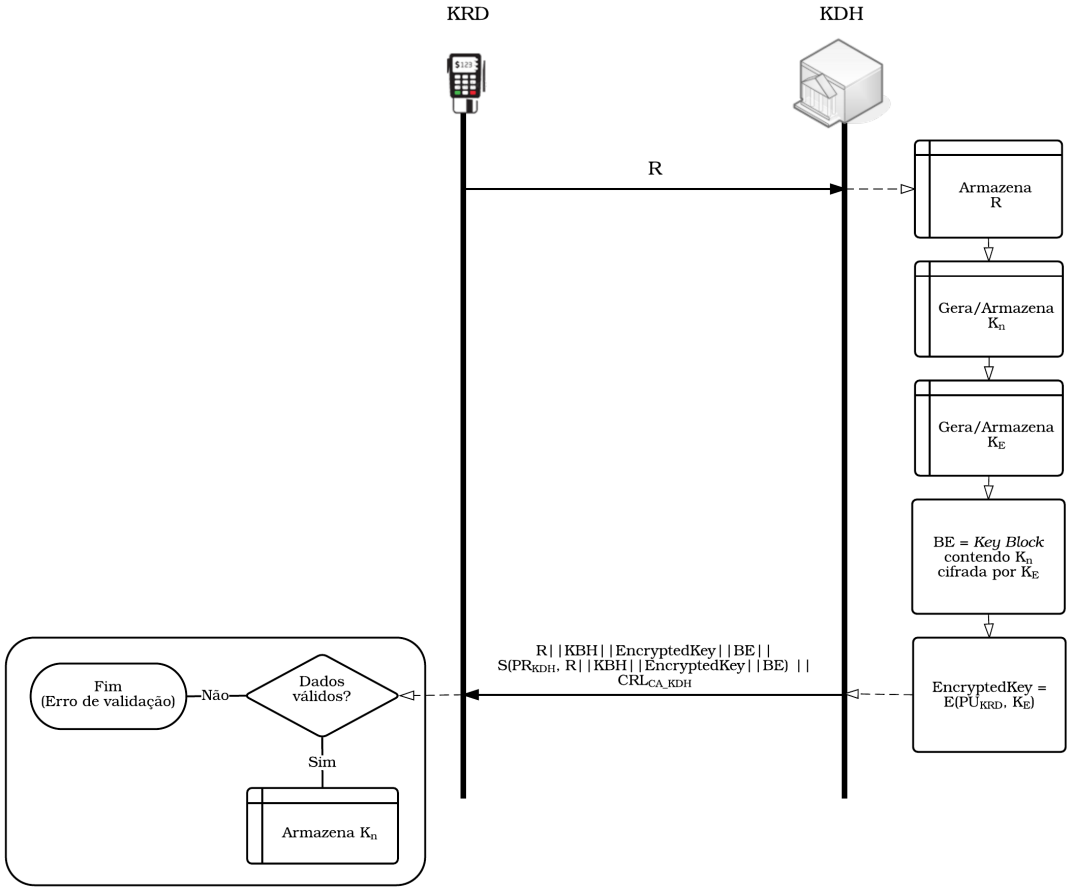

Figura 13: Protocolo TR34: Fase de transporte

O documento ASC X9 TR34 apresenta também uma fase de transporte da chave simétrica baseada em *timestamps*. Neste caso, esta fase é executada apenas num único fluxo de informação em vez de dois, como visto atrás. As validações do KRD são idênticas, com a diferença de validar se a *timestamp* TS recebida é recente. Os dados que o KDH enviaria neste caso seriam:

$$KBH \parallel EncryptedKey \parallel BE \parallel TS \parallel$$
$$S(PR_{KDH}, (KBH \parallel EncryptedKey \parallel BE \parallel TS)) \parallel CRL_{CA\_KDH}$$

Fase 3: Confirmação

Nesta fase final, o KRD envia ao KDH o KCV da chave que recebeu para que o KDH valide que K_n é coincidente em ambas as partes, verificando que o valor que recebeu é igual ao KCV calculado localmente.

A Figura 14 esquematiza esta fase de confirmação.

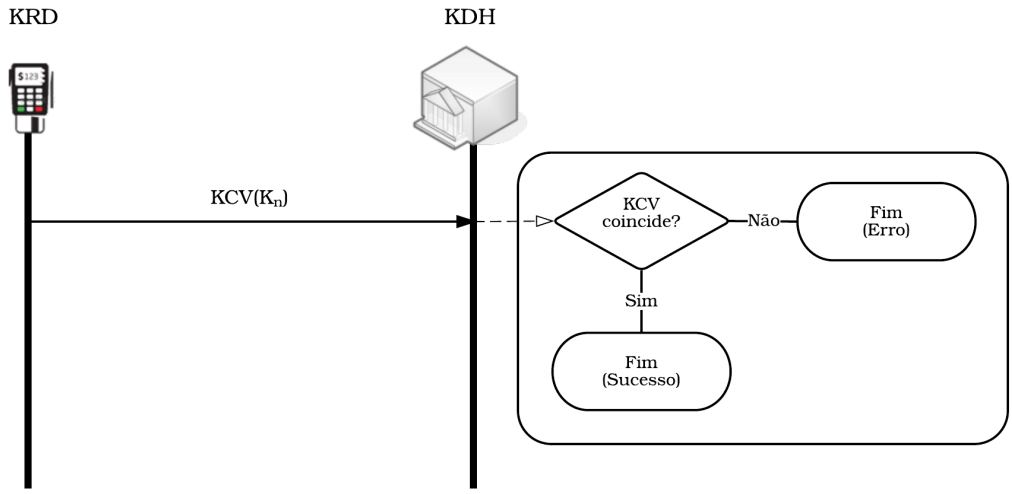

Figura 14: Protocolo TR34: Fase de confirmação

Análise

Utilizando o *Scyther* como ferramenta analisadora de protocolos de segurança é possível verificar o grau de segurança do método sugerido no documento ASC X9 TR34, nomeadamente quanto à confidencialidade, integridade e autenticidade da informação trocada.

Para isto, o *Scyther* será útil para avaliar a fase de **transporte** da chave simétrica a injetar no KRD. Em notação SPDL, esta fase define-se da seguinte forma:

```
usertype EphemeralKey;
usertype SharedKey;
usertype RoleId;
usertype KeyBlockHeader;
usertype KeyBlockVersion;
usertype CertificateRevocationList;

const KDHid: RoleId;
const KBH: KeyBlockHeader;
const Version: KeyBlockVersion;
const CRL: CertificateRevocationList;

hashfunction Hash;

protocol tr-34(KRD,KDH){
 role KRD {
  fresh R: Nonce;
  var Kn: SharedKey;
  var KE: EphemeralKey;

  send_1(KRD,KDH,R);
  recv_2(KDH,KRD,R,KBH,{KE}pk(KRD),{Version,KDHid,Kn,KBH}KE,{Hash(R,KBH,{KE}pk(KRD)
      ,{Version,KDHid,Kn,KBH}KE)}sk(KDH),CRL);

  claim(KRD,SKR,KE);
  claim(KRD,Secret,Kn);
  claim(KRD,Alive);
  claim(KRD,Weakagree);
  claim(KRD,Niagree);
  claim(KRD,Nisynch);
```

```
30     claim(KRD,Commit,KDH,R);
31    }
32
33    role KDH {
34     var R: Nonce;
35     fresh Kn: SharedKey;
36     fresh KE: EphemeralKey;
37
38     recv_1(KRD,KDH,R);
39     claim(KDH,Running,KRD,R);
40     send_2(KDH,KRD,R,KBH,{KE}pk(KRD),{Version,KDHid,Kn,KBH}KE,{Hash(R,KBH,{KE}pk(KRD)
           ,{Version,KDHid,Kn,KBH}KE)}sk(KDH),CRL);
41
42     claim(KDH,SKR,KE);
43     claim(KDH,Secret,Kn);
44     claim(KDH,Alive);
45     claim(KDH,Weakagree);
46     claim(KDH,Niagree);
47     claim(KDH,Nisynch);
48    }
49   }
```

Protocolo 3.1: Protocolo TR34

Executando a análise ao protocolo, o *Scyther* mostra que não existe qualquer problema em relação à confidencialidade dos termos KE e Kn (requisitos definidos nas linhas 24, 25, 42 e 43). Quanto à autenticação entre as partes, se por um lado a informação que o KDH envia ao KRD está autenticada pela assinatura RSA dos dados (ou seja, o KRD tem garantias de que está realmente a comunicar com o KDH), o inverso não se verifica: O KDH não tem garantias de estar a falar com o KRD. A Figura 15 apresenta os resultados devolvidos pelo *Scyther*.

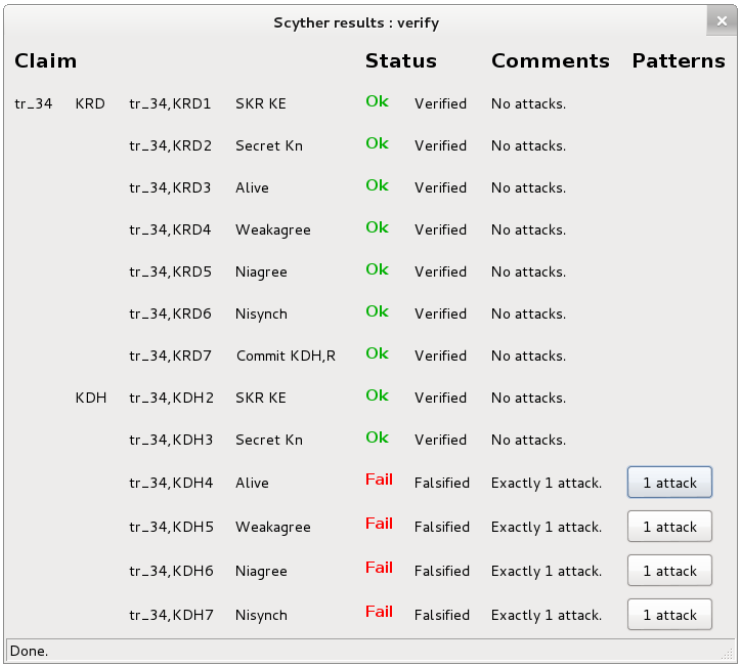

Figura 15: Análise ao Protocolo TR34 (*Scyther*)

De facto, no contexto do KDH, este não garante as propriedades de autenticação unilateral, autenticação mútua nem a integridade dos dados. Isto porque **qualquer adversário (*Eve*) pode enviar ao KDH um *nonce* R' sem que este se aperceba que o valor aleatório não veio realmente do KRD**. Se o intruso o fizer, não só quebrará a autenticação do protocolo como vai inviabilizar a receção com sucesso por parte do KRD que verificará que o nonce R (gerado por si) não coincidirá com R'. Adicionalmente, um adversário pode utilizar a informação que recebe do KDH e utilizá-la em ataques de criptoanálise com o objetivo de comprometer a chave privada do KRD.

Como é possível visualizar na Figura 15 existe exatamente um ataque que viola os requisitos de autenticação do KRD perante o KDH. Este ataque é semelhante para os quatro casos e é apresentado na Figura 16. O ataque consiste na introdução de um *nonce* R', conforme descrito atrás. Na parte superior da imagem é apresentado o *output* do *Scyther*, sendo que na parte inferior mostra-se o mesmo ataque de uma forma mais detalhada.

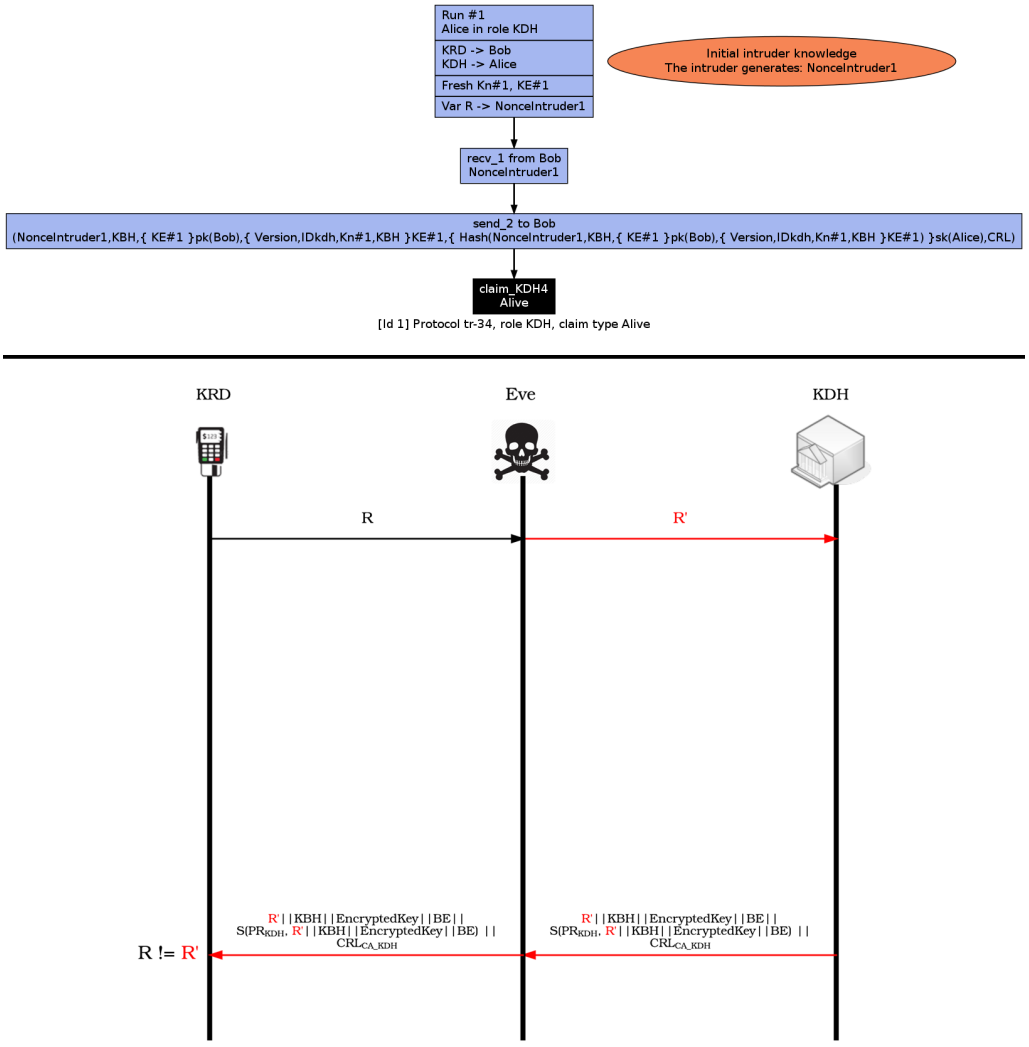

Figura 16: Ataque ao Protocolo TR34

Para além desta vulnerabilidade, o documento ASC X9 TR34 não especifica outros temas que não podem ser descurados aquando da definição do protocolo de injeção remota de chaves, sob pena de criar outras vulnerabilidades de segurança:

- **Como recebeu o KRD o seu par de chaves assimétricas?**

  A proposta TR34 inicia-se na fase de vinculação, o que pressupõe a existência *à priori* de um par de chaves assimétricas no KRD. Se a pré-instalação deste material criptográfico (e até a sua própria geração) não for feita corretamente poderá comprometer a chave privada do KRD, pondo em causa a confidencialidade necessária na fase de transporte da chave simétrica.
- **É possível garantir que a entidade que envia um certificado na fase de vinculação está mesmo na posse da respetiva chave privada?** A troca de certificados na fase inicial do protocolo garante apenas que estes são válidos. Por isto, é necessário garantir que quem envia um certificado tem realmente acesso à respetiva componente privada do par de chaves.
- **Qual o grau de confiança entre o KDH e o fabricante de KRDs que injeta previamente material criptográfico nos seus equipamentos?**

  Admitindo que o par de chaves do KRD foi instalado pelo seu fabricante, é necessário existir um grau de confiança entre o KDH e esse mesmo fabricante. Esta relação deverá ser construída de tal forma que não seja possível existir um fabricante desconhecido (e potencialmente malicioso) que consiga interagir com o KDH de modo a obter chaves simétricas indevidamente para os seus KRDs.

3.5 Conclusão do capítulo

Este capítulo apresentou as características de um POS, as suas chaves criptográficas e a forma como estas são atualmente carregadas pelo processador de transações. Introduziu também o conceito de injeção remota de chaves, cuja implementação deve reger-se pelas regras ditadas pela norma ANS X9.24-2. Foi analisada uma proposta de implementação de um protocolo desta natureza, ilustrado no documento ASC X9 TR34, que revelou alguns problemas de desenho.

Face às vulnerabilidades identificadas na implementação proposta, o próximo capítulo apresenta os detalhes de um protocolo seguro de injeção remota de chaves simétricas num POS, desde o momento ainda antes de sair de fábrica até ter instalado em si todas as chaves simétricas necessárias para o funcionamento na rede de processamento de transações financeiras.

Capítulo 4

Solução proposta

Não obstante as questões levantadas no final do capítulo anterior relacionadas com problemas na proposta TR34, esta representa um ponto de partida para a criação de um protocolo de injeção remota de chaves simétricas mais robusto do que o apresentado nesse documento. Desta forma, este capítulo detalha um protocolo baseado no TR34, não só cobrindo as suas vulnerabilidades mas também apresentando todo o processo de injeção de chaves, que se inicia desde logo no fabricante do POS. Este último desempenha um papel importante no protocolo cujas ações não foram detalhadas no documento ASC X9 TR34.

Esta solução proposta é analisada formalmente com recurso ao *Scyther*, sendo também avaliada a sua praticabilidade validando se o protocolo está desenhado de acordo com um conjunto de objetivos reunidos num documento também concretizado pela ASC: O **ASC X9 TR39**.

4.1 Análise de requisitos

Da investigação feita no início deste trabalho resultou uma análise de requisitos que o protocolo de injeção remota de chaves simétricas deve cumprir. Estes requisitos são listados de seguida.

4.1.1 Requisitos de criptografia simétrica

1. **Qualquer chave simétrica deverá ser gerada num componente seguro.**

   - Todas as chaves simétricas (a injetar ou auxiliares ao protocolo) deverão ser geradas com recurso a um Hardware Security Module (HSM) e nunca por *software* ou outro método. Estes componentes seguros são construídos para gerar (e armazenar) chaves criptográficas, utilizando mecanismos comprovadamente robustos do ponto de vista de segurança (como geradores de números aleatórios, entre outros) [24].

2. **Chaves simétricas deverão apenas existir no interior de um componente seguro (HSM, Trusted Platform Module (TPM), etc.) ou, se fora de um destes equipamentos, cifradas por meio de criptografia simétrica ou assimétrica.**

   - Estas chaves devem estar sempre guardadas de forma segura e nunca poderão estar em claro em nenhum momento no seu ciclo de vida. Quando chaves simétricas estiverem fora do componente seguro estas devem estar cifradas por meio de criptografia simétrica ou assimétrica, sendo que a operação de cifra tem de ser efetuada dentro do HSM.

3. **Uma chave simétrica deve ser apenas utilizada para um único fim.**

   - Tanto as chaves simétricas a injetar remotamente como as que possam existir para auxiliar o protocolo deverão ser apenas usadas com uma só finalidade (por exemplo, uma chave utilizada para cifra de informação não poderá ser utilizada para calcular MACs).

4.1.2 Requisitos de criptografia assimétrica

4. **Qualquer par de chaves deverá ser gerado num componente seguro.**

   - À semelhança de chaves simétricas, pares de chaves assimétricas deverão ser gerados num HSM e nunca por outro método.

5. **Cada interveniente deve ter o seu próprio par de chaves.**

   - Nenhum par de chaves assimétricas deverá ser partilhado por dois ou mais intervenientes.

6. **Chaves privadas deverão apenas existir no interior de um componente seguro ou, se fora, cifradas por meio de criptografia simétrica.**

   - Tal como as chaves simétricas, estas devem estar armazenadas de forma segura, seja dentro de um componente seguro ou cifradas por criptografia simétrica, sendo que o tamanho da chave simétrica que cifra uma chave privada deve ser proporcional ao tamanho desta última.

7. **Chaves privadas deverão ser utilizadas apenas para decifrar ou para assinaturas digitais, mas não para ambas as situações.**

   - À imagem das chaves simétricas, uma chave privada deverá ser utilizada apenas para um único propósito.

8. **Uma chave privada deve apenas existir nos locais em que esta será usada.**

   - Cada chave privada deverá existir no menor número de locais possível, sendo que idealmente deverá apenas existir no local onde será utilizada. Caso um interveniente receba uma chave privada de outro (por impossibilidade de geração de par de chaves assimétricas do primeiro) esta deve ser imediatamente eliminada depois de enviada.

9. **Chaves públicas deverão apenas existir nas mesmas formas que uma chave privada pode existir ou num certificado digital.**

- Ainda que a sua confidencialidade não necessite de ser preservada, deve-se garantir a integridade e autenticidade das chaves públicas a utilizar no protocolo. Por este motivo, estas devem ser mantidas tal como uma chave privada ou, em alternativa, encapsuladas num certificado digital.

10. **Chaves públicas deverão ser utilizadas apenas para cifrar ou para validação de assinaturas digitais, mas não para ambas as situações.**

    - Também como as chaves privadas e as chaves simétricas, estas devem ser apenas usadas para um único propósito.

11. **À exceção de eventuais Autoridades de Certificação (CAs) de topo existentes no protocolo, nenhuma outra entidade deverá ter um certificado digital auto-assinado.**

    - Qualquer certificado existente no protocolo deverá ser assinado por uma CA confiável, não podendo existir qualquer certificado auto-assinado (exceto certificados de CAs de topo que, por definição, assinam o seu próprio certificado digital).

4.1.3 Requisitos do protocolo

12. **Quaisquer entidades intervenientes no protocolo deverão autenticar-se mutuamente.**

    - Os intervenientes no protocolo deverão estar mutuamente autenticados antes de iniciarem qualquer troca de mensagens. Esta autenticação deve ser feita tanto pela apresentação dos seus certificados digitais, como pela prova de posse da respetiva chave privada.

13. **Deverá ser validada a autenticidade de qualquer certificado digital.**

    - Deverá ser validada a integridade e autenticidade de qualquer certificado envolvido no protocolo, não devendo ser utilizada uma chave pública que não tenha sido devidamente autenticada. A validação do conteúdo dos certificados também deverá ser efetuada. Nomeadamente, deve ser verificado o nome do seu proprietário, a sua data de expiração, e usos permitidos para a chave pública associada ao certificado. Também é necessário verificar se o certificado é assinado digitalmente pela CA expectável e que a assinatura é válida.

14. **O protocolo deverá ter mecanismos de prevenção contra ataques "homem-no-meio" e ataques de repetição.**

    - O desenho do protocolo deve ter em conta formas de prevenir que um adversário tenha a possibilidade de se interpor na comunicação e comprometer ou alterar as chaves a injetar, bem como tentar repetir uma comunicação anterior de modo a que sejam instaladas chaves simétricas enviadas no passado.

15. **Deve ser preservada a confidencialidade, integridade, autenticidade e não-repúdio de envio das chaves simétricas a injetar.**

    - As chaves simétricas a partilhar entre as partes deve manter-se apenas do co-

nhecimento das mesmas. Deve também ser garantido que estas não foram alteradas por um adversário e o seu envio não poderá ser negado por quem as transmitiu.

4.2 Entidades envolvidas

A execução do protocolo depende de três entidades cujos objetivos são os seguintes:

KDH
O Key Distribution Host (KDH): A entidade distribuidora de chaves simétricas para os POSs de qualquer fabricante. É, em simultâneo, a entidade processadora de transações. Gere também a CA *root* da solução proposta, CA_{KDH}. Esta assina o certificado das CAs subordinadas mantidas pelos fabricantes, bem como o certificado associado ao par de chaves utilizado para assinar digitalmente os dados a enviar ao POS.

P
Um dado POS que receberá chaves simétricas a partir de um ponto central de gestão de chaves criptográficas.

F
O fabricante do POS P. É responsável por providenciar um par de chaves assimétricas nos POSs que fabrique, assinados pela sua sub-CA CA_F.

4.3 Certificação de fabricantes

Dado que os fabricantes de POSs desempenham um papel fulcral na fase inicial do protocolo, será necessário estabelecer uma relação de confiança entre eles e o KDH. A sua função passará por providenciar aos POSs um par de chaves assimétricas antes de saírem das suas instalações para que o protocolo de carregamento remoto de chaves se possa concretizar.

Este modelo de confiança proposto, designado por **Two-Party Model** [2], caracteriza-se pela existência de duas CAs: O KDH atua como CA de topo, sendo que os fabricantes de terminais assumem o papel de CAs subordinadas da anterior.

A Figura 17 exemplifica esta hierarquia relacional tomando como exemplo dois fabricantes distintos. Desta forma existe uma clara segregação de material criptográfico entre fabricantes, não existindo partilha de chaves entre nenhuma das entidades. Isto garante que o comprometimento de uma chave ao nível do domínio de um fabricante não afeta negativamente outras entidades homólogas.

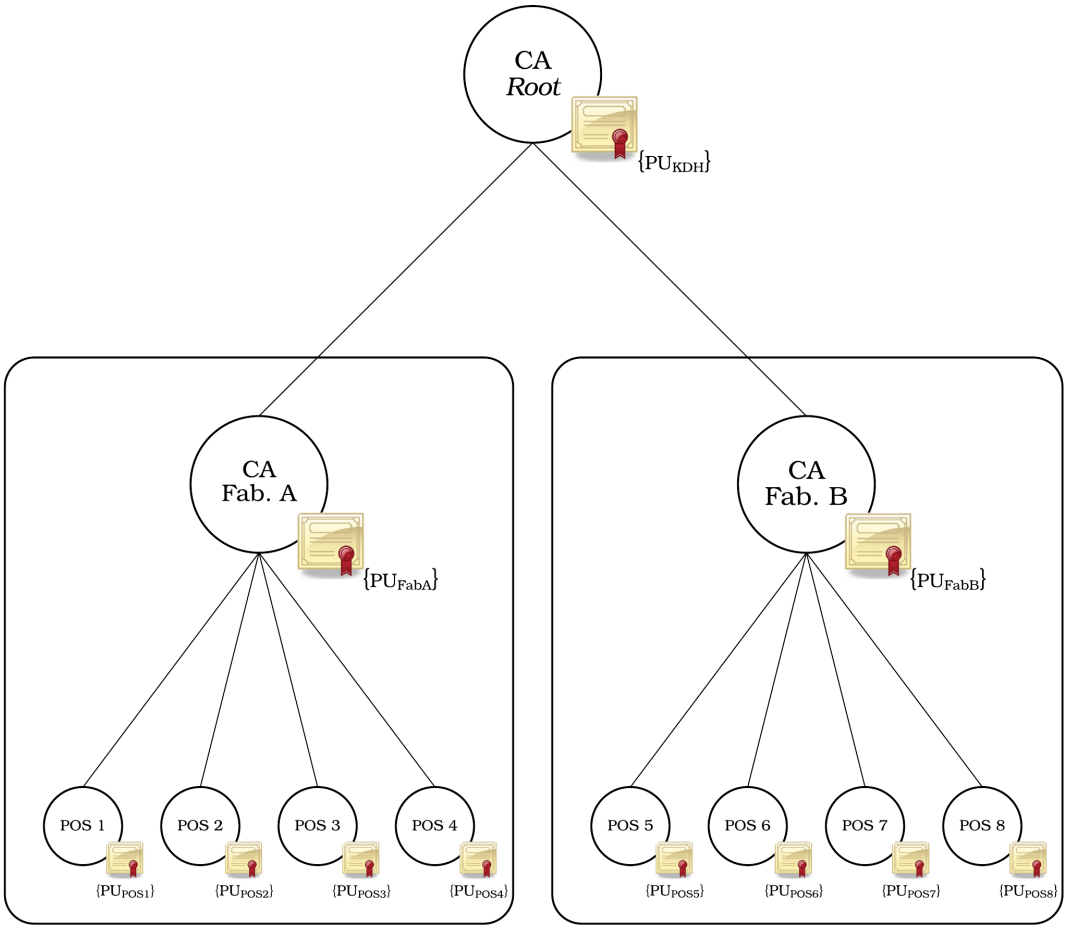

Figura 17: Modelo de confiança entre o KDH e os fabricantes de POSs

4.4 Assinatura de certificado da CA do fabricante

Este processo de estabelecimento da relação de confiança entre o KDH e F, deve desenrolar-se do seguinte modo:

1. F gera um par de chaves assimétricas no seu HSM, PU_F e PR_F.
2. F gera também um pedido de assinatura de um certificado contendo a sua chave pública, enviando-o à CA_{KDH}. Este pedido de assinatura deverá ser formatado de acordo com a norma PKCS #10 [18].
3. A CA_{KDH} assina o pedido, gerando $Cert\{PU_F\}$.
4. A CA_{KDH} envia a F o seu certificado devidamente assinado.
5. F guarda o certificado que recebeu da CA_{KDH} no seu HSM, ficando assim a CA_F um Sub-CA da CA_{KDH}.

A Figura 18 apresenta este procedimento esquematicamente.

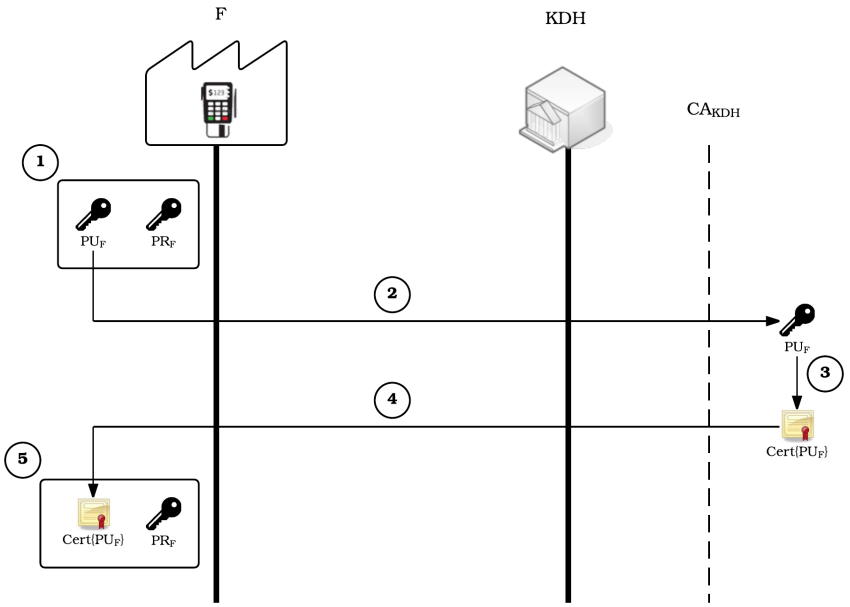

Figura 18: Estabelecimento da relação de confiança entre um fabricante, F, e o KDH

De notar que o passo 2 deste procedimento é um dos pontos críticos do processo pois caso um atacante consiga que um par de chaves controlado por si seja assinado pelo KDH, este passará a ser um fabricante malicioso que pode fabricar POSs comprometidos. Consequentemente, o KDH deverá fazer prova de identidade do fabricante, bem como receber o pedido de assinatura de certificado por um canal alternativo (por exemplo: num dispositivo de armazenamento via correio).

4.5 Geração de par de chaves de assinatura RSA

Tal como na proposta TR34, o protocolo final que será proposto mais adiante protegerá a autenticidade e integridade da informação a transportar com base em assinaturas RSA. Assim, o KDH gera um par de chaves único por fabricante e assinado pela sua CA *root*. O certificado devidamente assinado será enviado ao respetivo fabricante apenas para que este o instale nos seus POSs para posterior utilização. O estabelecimento desta chave de assinatura faz-se da seguinte maneira:

1. O KDH gera um par de chaves no seu HSM, PU_{signer_F} e PR_{signer_F}.
2. O KDH gera um pedido de assinatura de certificado PKCS #10 que será assinado pela sua CA_{KDH}, dando origem ao certificado $Cert\{PU_{signer\_F}\}$.
3. O KDH envia $Cert\{PU_{signer\_F}\}$ a F para que este o injete em cada um dos seus terminais.
4. F guarda o certificado recebido de forma segura no seu HSM.

O facto de este par de chaves ser único por fabricante tem o intuito de garantir que, caso este par seja comprometido não afetará POSs de outros fabricantes, dado que cada um destes pares só assina dados a enviar aos terminais de um fabricante. Garante-se, mais uma vez, a segregação do material criptográfico das várias entidades envolvidas no protocolo. Este par de chaves é importante para garantir que não existem chaves públicas ou privadas para múltiplos fins porque, não obstante o KDH já ter um par de chaves RSA utilizado no contexto da CA_{KDH}, este só pode ser utilizado para assinar certificados digitais e não outro tipo de dados.

A Figura 19 ilustra este processo.

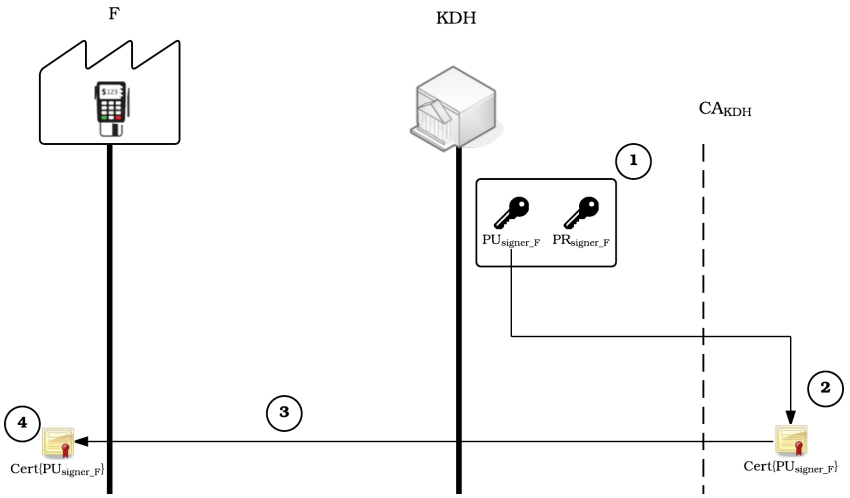

Figura 19: Estabelecimento de par de chaves de assinatura RSA, único por fabricante

4.6 Pré-instalação de par de chaves assimétricas num POS

Com o seu par de chaves assinado pela CA *root* do protocolo, o fabricante, por meio da sua CA_F, está em condições de gerar, assinar e instalar em cada um dos seus POSs um par de chaves assimétricas, único por equipamento. Esta instalação de chaves assimétricas em cada terminal, ainda no seu local de fabrico, deverá ser executada da seguinte forma:

1. F gera um par de chaves assimétricas, PU_P e PR_P, no seu HSM.
2. A CA_F assina o certificado contendo a chave pública de P, dando origem ao certificado $Cert\{PU_P\}$.
3. F instala a chave privada e o certificado devidamente assinado em P, bem como o certificado da chave pública do par de chaves assimétricas para assinatura de dados por parte do KDH, $Cert\{PU_{signer\_F}\}$.
4. P guarda todo este material criptográfico de forma segura, no seu TPM.

Esta pré-injeção de material criptográfico nos POSs por parte do seu fabricante deve ocorrer numa sala segura, de modo a minimizar o risco de comprometimento da chave privada PR_P. Adicionalmente, esta chave deve ser prontamente destruída no HSM do fabricante após ter sido instalada no respetivo POS.

A Figura 20 esquematiza este procedimento.

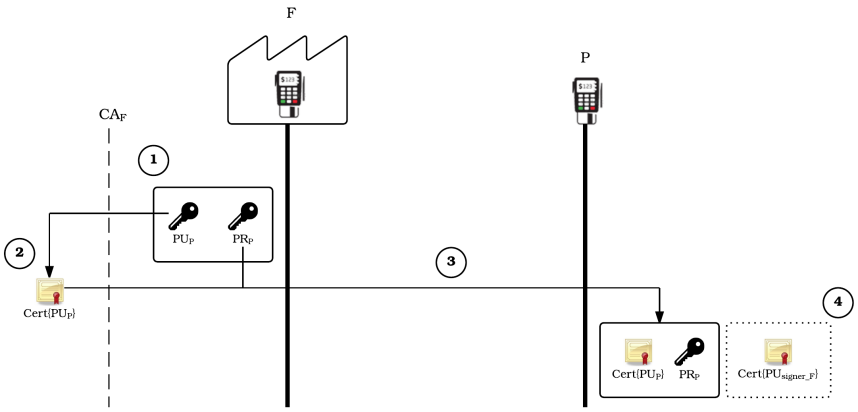

Figura 20: Pré-instalação de elementos assimétricos em P

4.7 Protocolo de transporte de chaves simétricas

A solução proposta divide-se em três fases baseadas nas mesmas que o protocolo TR34, porém, com alguns elementos adicionais. Os três momentos deste protocolo apresentam-se de seguida.

4.7.1 Fase 1: Autenticação

Esta primeira fase caracteriza-se não só pela troca de certificados digitais entre P e o KDH (tal como visto no TR34) mas também pela necessidade de se fazer prova de que o outro interveniente está realmente na posse da respetiva chave privada. Esta fase executa-se nos seguintes passos:

1. P envia ao KDH o seu certificado digital, $Cert\{PU_P\}$, e um *nonce* R_P.
2. O KDH verifica $Cert\{PU_P\}$. Nomeadamente, valida que:
   (a) O nome do proprietário incluído no certificado é o número de série (ID_P) de um POS conhecido[1].
   (b) O certificado não está expirado.

[1]Por exemplo, o técnico de instalação do POS no local pode informar o KDH desta ação e este insere o número de série do respetivo terminal numa base de dados criada para esta validação. Só após esta interação é que o processo de injeção remota de chaves pode dar início.

(c) O certificado foi assinado pelas CAs expectáveis (CA_F, que por sua vez é assinada pela CA_{KDH}).

(d) A sua assinatura digital é válida.

3. Caso a validação anterior seja feita com sucesso, O KDH envia a P o certificado digital $Cert\{PU_{signer\_F}\}$, R_P assinado pela respetiva componente privada do referido certificado, e um *nonce* R_{KDH} cifrado pela chave pública de P.
4. P valida $Cert\{PU_{signer\_F}\}$. Nomeadamente, valida que o certificado digital recebido é o mesmo que foi injetado pelo fabricante no momento de pré-instalação de material criptográfico. Igualmente, P valida a assinatura de R_P e, caso seja válida, responde ao KDH enviando-lhe R_{KDH} decifrado.
5. Caso o KDH confirme que R_{KDH} é o correto, a fase de autenticação termina com sucesso, com ambas as partes a guardarem os certificados recebidos para utilização na fase seguinte.

A Figura 21 esquematiza esta fase.

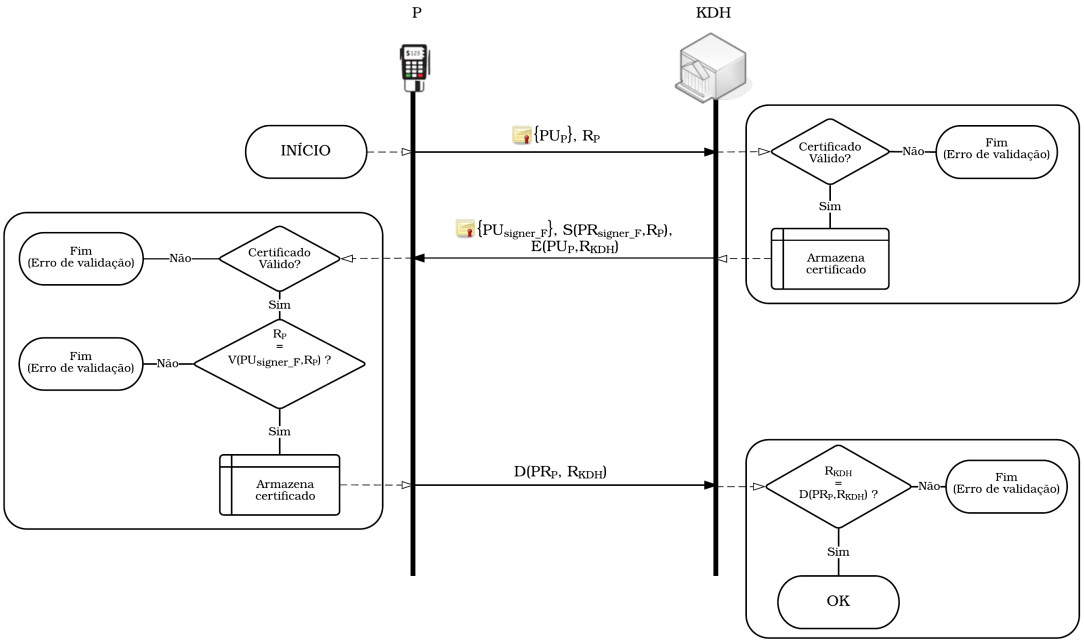

Figura 21: Fase 1 do protocolo proposto: Autenticação

4.7.2 Fase 2: Transporte

Após ambas as partes se terem autenticando mutuamente estão garantidas as condições para se fazer o transporte das chaves simétricas que um POS necessita: A Terminal Master Key (TMK) e as três Terminal Working Keys (TWKs): K_{TMK}, K_{CIF}, K_{PIN} e K_{MAC}. Por motivos de simplificação, estas chaves serão transportadas em forma de criptograma

(isto é, cifradas por outra chave simétrica) e não numa estrutura de dados como o *key block* TR34.

Esta parte do protocolo executa-se do seguinte modo:

1. O KDH começa por gerar um *nonce*, R_1, e uma chave simétrica de sessão, K_E. O KDH envia a P R_1 bem como K_E cifrada pela chave pública de P. Envia, igualmente, a assinatura digital destes dados também ela cifrada por K_E.
2. P recebe estes dados e valida a sua assinatura. Caso seja válida, P guarda K_E e gera um segundo *nonce*, R_2, enviando-o ao KDH concatenado com ID_P, sendo isto cifrado pela chave efémera K_E.
3. Com a informação recebida, o KDH decifra os dados e verifica se a parte final coincide com o valor esperado de ID_P. Em caso afirmativo, são geradas a K_{TMK}, K_{CIF}, K_{PIN} e K_{MAC}, e são enviadas a P juntamente com R_2 recebido no ponto anterior, bem como a assinatura dos dados. Tudo isto é cifrado pela chave de sessão K_E.
4. P recebe esta informação e após validar a assinatura digital, pode decifrar as chaves com K_E e guardá-las no seu TPM.

Este procedimento está ilustrado na Figura 22.

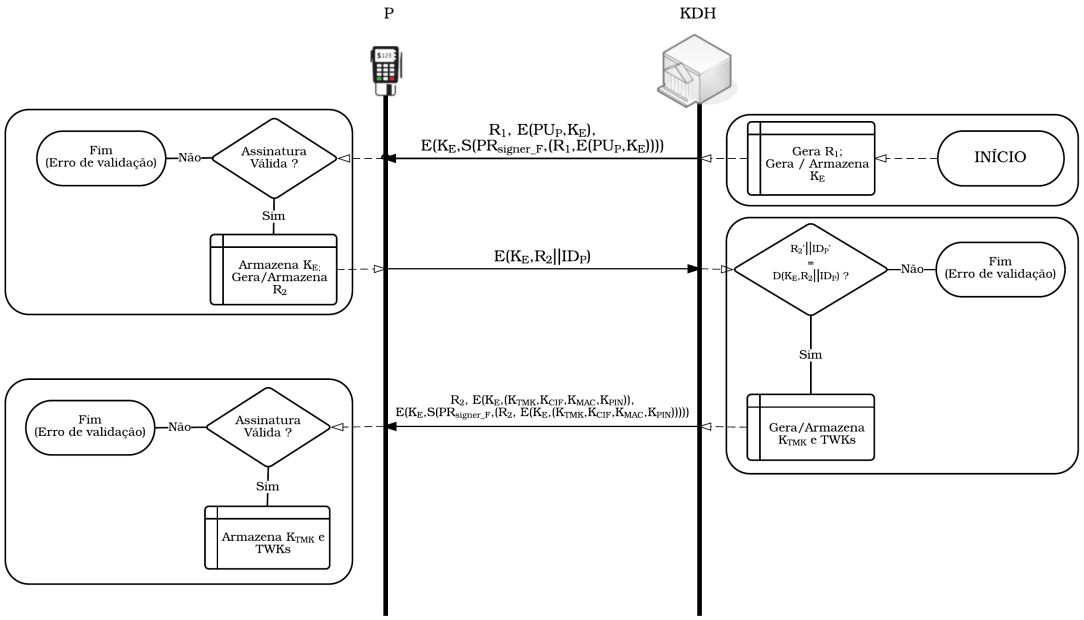

Figura 22: Fase 2 do protocolo proposto: Transporte

4.7.3 Fase 3: Confirmação

A fase final do protocolo tem o objetivo de ser dado ao KDH a garantia de que P recebeu as chaves corretamente. Para tal:

1. P envia ao KDH, cifrado por K_E, um *nonce* R_3 e os KCVs das quatro chaves injetadas: $KCV(K_{TMK})$, $KCV(K_{CIF})$, $KCV(K_{MAC})$ e $KCV(K_{PIN})$.
2. O KDH verifica que os KCVs são os esperados e, em caso positivo, notifica P do sucesso da injeção de chaves com essa informação juntamente com o *nonce* R_3, tudo isto cifrado por K_E. É também enviada a assinatura digital desta informação cifrada pela chave efémera. Caso os KCVs não coincidam, P é igualmente notificado com essa informação.
3. P termina o protocolo confirmando o sucesso (ou insucesso) da execução do mesmo, e que R_3 tem o valor esperado.

Esta fase de confirmação é apresentada na Figura 23.

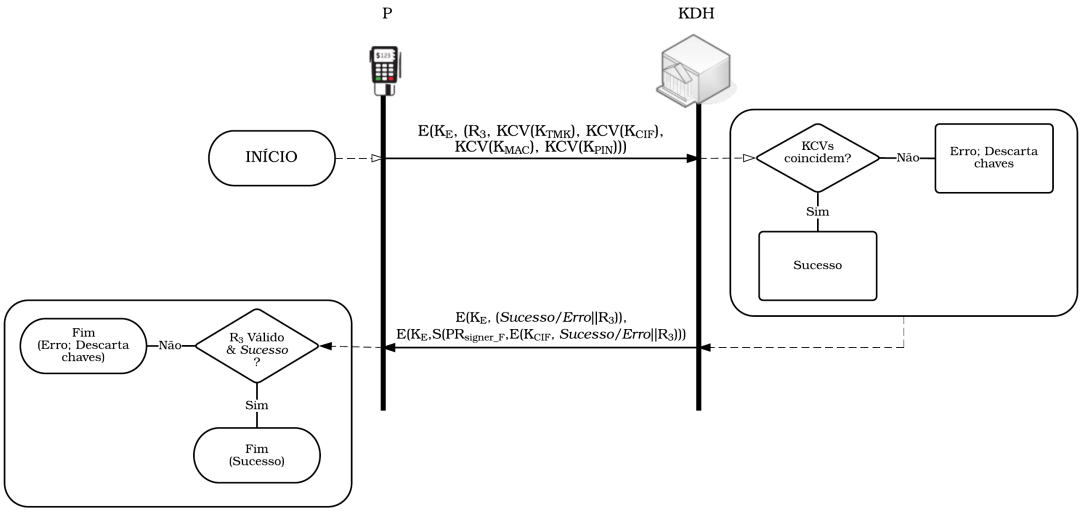

Figura 23: Fase 3 do protocolo proposto: Confirmação

Com este desenho de protocolo em mente, é necessário analisá-lo de modo a aferir a sua praticabilidade e o seu nível de segurança face aos requisitos levantados inicialmente. A secção seguinte apresenta esta análise e os respetivos resultados.

4.8 Análise e resultados

Esta secção divide-se em duas componentes de avaliação: Por um lado, analisa-se o protocolo numa vertente formal com recurso ao *Scyther*. Por outro, apresenta-se uma análise do ponto de vista regulatório com recurso a um documento denominado **ASC X9 TR39** [3].

Esta guia, publicado pela ASC (tal como a ANS X9.24-2 e o ASC X9 TR34), apresenta uma lista de verificação que, entre outros objetivos, deve servir de avaliação a qualquer protocolo de gestão de chaves simétricas por meio de criptografia assimétrica no ramo financeiro, sob pena de as organizações que não a cumpram entrarem em incumprimento legal e, mais importante, executarem um protocolo de injeção remota de chaves simétricas com vulnerabilidades de segurança.

No final, são também apresentados alguns aspetos relevantes do protocolo de modo a entender a inclusão de determinados elementos na solução proposta.

4.8.1 Análise via *Scyther*

Abaixo apresentam-se as fases de transporte e de confirmação da solução apresentada no capítulo anterior em notação SPDL e que é passada como *input* ao *Scyther*. A fase de autenticação não é analisada por esta ferramenta por não ter *claims* a validar, seja de confidencialidade, autenticação, integridade ou ordem.

```
usertype EphemeralKey;
usertype SharedKey;
usertype RoleId;
usertype KCV;
usertype Result;

const IDp: RoleId;
hashfunction Hash;

protocol enhancedTR-34(P,KDH){
    role KDH {
        fresh R1: Nonce;
        fresh KE: EphemeralKey;
        fresh Ktmk,Kcif,Kmac,Kpin: SharedKey;
        fresh res: Result;
        var KCVktmk,KCVkcif,KCVkmac,KCVkpin: KCV;
        var R2,R3: Nonce;

        send_1(KDH,P,R1,{KE}pk(P),{{Hash(R1,{KE}pk(P))}sk(KDH)}KE);
        recv_2(P,KDH,{R2,IDp}KE);
        claim(KDH,Running,P,R2);
        send_3(KDH,P,R2,{Ktmk,Kcif,Kmac,Kpin}KE,{{Hash(R2,{Ktmk,Kcif,Kmac,Kpin}KE)}
            sk(KDH)}KE);
        recv_4(P,KDH,{R3,KCVktmk,KCVkcif,KCVkmac,KCVkpin}KE);
        claim(KDH,Running,P,R3);
        send_5(KDH,P,{res,R3}KE,{{Hash({res,R3}KE)}sk(KDH)}KE);

        claim(KDH,SKR,KE);
        claim(KDH,Secret,Ktmk);
        claim(KDH,Secret,Kcif);
        claim(KDH,Secret,Kmac);
        claim(KDH,Secret,Kpin);
        claim(KDH,Secret,res);
        claim(KDH,Alive);
        claim(KDH,Weakagree);
        claim(KDH,Niagree);
        claim(KDH,Nisynch);
        claim(KDH,Commit,P,R1);
    }
    role P {
        fresh R2,R3: Nonce;
        fresh KCVktmk,KCVkcif,KCVkmac,KCVkpin: KCV;
        var KE: EphemeralKey;
```

```
43          var R1: Nonce;
44          var Ktmk,Kcif,Kmac,Kpin: SharedKey;
45          var res: Result;
46
47          recv_1(KDH,P,R1,{KE}pk(P),{{Hash(R1,{KE}pk(P))}sk(KDH)}KE);
48          claim(P,Running,KDH,R1);
49          send_2(P,KDH,{R2,IDp}KE);
50          recv_3(KDH,P,R2,{Ktmk,Kcif,Kmac,Kpin}KE,{{Hash(R2,{Ktmk,Kcif,Kmac,Kpin}KE)}
                sk(KDH)}KE);
51          send_4(P,KDH,{R3,KCVktmk,KCVkcif,KCVkmac,KCVkpin}KE);
52          recv_5(KDH,P,{res,R3}KE,{{Hash({res,R3}KE)}sk(KDH)}KE);
53
54          claim(P,SKR,KE);
55          claim(P,Secret,Ktmk);
56          claim(P,Secret,Kcif);
57          claim(P,Secret,Kmac);
58          claim(P,Secret,Kpin);
59          claim(P,Secret,res);
60          claim(P,Alive);
61          claim(P,Weakagree);
62          claim(P,Niagree);
63          claim(P,Nisynch);
64          claim(P,Commit,KDH,R2);
65          claim(P,Commit,KDH,R3);
66      }
67  }
```

Protocolo 4.1: Protocolo proposto

Resultados

O resultado da avaliação da ferramenta *Scyther* apresentam-se na Figura 24.

Scyther results : verify

Claim				Status		Comments
enhancedTR_34	KDH	enhancedTR_34,KDH3	SKR KE	Ok	Verified	No attacks.
		enhancedTR_34,KDH4	Secret Ktmk	Ok	Verified	No attacks.
		enhancedTR_34,KDH5	Secret Kcif	Ok	Verified	No attacks.
		enhancedTR_34,KDH6	Secret Kmac	Ok	Verified	No attacks.
		enhancedTR_34,KDH7	Secret Kpin	Ok	Verified	No attacks.
		enhancedTR_34,KDH8	Secret res	Ok	Verified	No attacks.
		enhancedTR_34,KDH9	Alive	Ok	Verified	No attacks.
		enhancedTR_34,KDH10	Weakagree	Ok	Verified	No attacks.
		enhancedTR_34,KDH11	Niagree	Ok	Verified	No attacks.
		enhancedTR_34,KDH12	Nisynch	Ok	Verified	No attacks.
		enhancedTR_34,KDH13	Commit P,R1	Ok	Verified	No attacks.
	P	enhancedTR_34,P2	SKR KE	Ok	Verified	No attacks.
		enhancedTR_34,P3	Secret Ktmk	Ok	Verified	No attacks.
		enhancedTR_34,P4	Secret Kcif	Ok	Verified	No attacks.
		enhancedTR_34,P5	Secret Kmac	Ok	Verified	No attacks.
		enhancedTR_34,P6	Secret Kpin	Ok	Verified	No attacks.
		enhancedTR_34,P7	Secret res	Ok	Verified	No attacks.
		enhancedTR_34,P8	Alive	Ok	Verified	No attacks.
		enhancedTR_34,P9	Weakagree	Ok	Verified	No attacks.
		enhancedTR_34,P10	Niagree	Ok	Verified	No attacks.
		enhancedTR_34,P11	Nisynch	Ok	Verified	No attacks.
		enhancedTR_34,P12	Commit KDH,R2	Ok	Verified	No attacks.
		enhancedTR_34,P13	Commit KDH,R3	Ok	Verified	No attacks.

Done.

Figura 24: Resultados da análise à solução proposta (*Scyther*)

O *Scyther* indica que todos os requisitos (*claims*) a testar foram validados com sucesso, sem qualquer ataque detetado pela ferramenta. Nomeadamente, garantiu-se a integridade, autenticidade e ordem da troca de mensagens, bem como a confidencialidade da chave de sessão K_E, das quatro chaves injetadas em P e ainda da notificação do resultado da execução do protocolo.

4.8.2 Avaliação via ASC X9 TR39

A segunda avaliação passível de se efetuar utiliza como base o documento ASC X9 TR39. Entre outros temas, este guia apresenta um conjunto de objetivos que qualquer organização do ramo financeiro deve atingir caso utilize em algum contexto criptografia assimétrica para gerir chaves simétricas, como é o caso da solução proposta neste trabalho.

A Tabela 4 apresenta os 17 objetivos mandatórios definidos no ASC X9 TR39 no contexto de gestão de chaves simétricas por meio de criptografia assimétrica. Para cada objetivo é necessário indicar se este é cumprido, não cumprido, ou não-aplicável. Adicionalmente, justifica-se cada uma das respostas.

Objetivo	**Sim**	**Não**	**N/A**	**Justificação**
Autenticação das chaves públicas: É assegurado que todas as chaves públicas são autenticadas antes de serem utilizadas.	✓			Na fase inicial do protocolo ambos os intervenientes validam os seus certificados de chave pública.
Utilização de algoritmos de acordo de chaves simétricas: Se for utilizado qualquer algoritmo de acordo de chaves simétricas, este deve ser um algoritmo aprovado pela ASC.			✓	Não existem algoritmos de acordo de chaves envolvidos no protocolo.
Utilização de algoritmos de transporte bilateral de chaves simétricas: Se for utilizado qualquer algoritmo de transporte bilateral de chaves simétricas, a função que funde as duas componentes numa só chave simétrica está de acordo com as normas da ASC.			✓	Não existem algoritmos de transporte bilateral de chaves envolvidos no protocolo.
Utilização de algoritmos de transporte unilateral de chaves simétricas: Se for utilizado qualquer algoritmo de transporte unilateral de chaves simétricas, este está de acordo com as normas da ASC.	✓			O algoritmo de transporte unilateral de chaves simétricas é baseado na proposta TR34, que por sua vez baseia-se na norma ANS X9.24-2.
Utilização do modelo de confiança *Two-party model*: Se for utilizado o modelo de confiança *two-party model* entre entidades, este está desenhado de acordo com o estipulado pelas normas da ASC.	✓			O modelo de confiança proposto baseia-se no *two-party model* descrito na norma ANS X9.24-2.
Utilização do modelo de confiança *Three-party model*: Se for utilizado o modelo de confiança *three-party model* entre entidades, este está desenhado de acordo com o estipulado pelas normas da ASC.			✓	Este modelo de confiança não é utilizado.

Utilização do modelo de confiança *Prior Trust*: Se for utilizado o modelo de confiança *prior trust* entre entidades, este está desenhado de acordo com o estipulado pelas normas da ASC.			✓	Este modelo de confiança não é utilizado.
Existência de chaves simétricas: Chaves simétricas só existem dentro de um componente seguro ou, se fora, cifradas por meio de criptografia simétrica ou assimétrica.	✓			As chaves simétricas são geradas no HSM do KDH e transportadas cifradas até P que, por sua vez, as guarda no seu componente seguro.
Existência de chaves privadas: Chaves privadas só existem dentro de um componente seguro ou, se fora, cifradas por meio de criptografia simétrica.	✓			Todas as chaves privadas são guardadas num componente seguro, tanto do lado do KDH como em P ou no seu fabricante F.
Existência de chaves públicas: Chaves públicas só existem nas formas que uma chave privada pode existir ou associadas a um certificado digital.	✓			Todas as chaves públicas estão contidas num certificado digital X.509.
Manipulação indevida de chaves simétricas: Existem mecanismos que detetem e/ou previnam modificação, substituição ou repetição indevida de uma chave simétrica transportada.	✓			As chaves simétricas são assinadas digitalmente pelo KDH, prevenindo a sua manipulação. No seu envio, os *nonces* existentes previnem ataques de repetição.
Par de chaves de um POS: O par de chaves a utilizar num POS, se gerado fora do mesmo, só poderá existir no HSM que o gerou e deverá ser eliminado do mesmo imediatamente após ter sido transferido para o POS. A integridade e autenticidade da chave privada deve ser mantida.	✓			O par de chaves de P é gerado e assinado no HSM de F. O mesmo equipamento insere este par em P e elimina-o imediatamente.
Pares de chaves únicos: Todos os pares de chaves existentes são únicos por entidade.	✓			No protocolo, não existem pares de chaves partilhados por qualquer entidade.
Geração de pares de chaves: Qualquer par de chaves assimétricas é gerado num componente seguro. É garantida a confidencialidade da chave privada e integridade da chave pública.	✓			Todos os pares de chaves são gerados num componente seguro. A chave privada é mantida num componente seguro, assegurando a sua confidencialidade e integridade. A chave pública é mantida num certificado X.509.
Utilização de pares de chaves: Chaves privadas são utilizadas apenas para decifra de informação ou assinaturas digitais, mas nunca para ambos. Chaves públicas são utilizadas apenas para cifra de informação ou validação de assinaturas, mas nunca para ambos.	✓			No protocolo proposto, nenhuma chave assimétrica é utilizada para mais do que uma função.
Autenticação do POS: Qualquer POS que necessite de chaves simétricas deve autenticar-se previamente perante a entidade que as injetará.	✓			P autentica-se perante o KDH enviando-lhe o seu certificado digital e fazendo prova de posse da respetiva chave privada, por meio de um mecanismo de "desafio-resposta".
Autenticação da entidade distribuidora de chaves simétricas: A entidade que distribui chaves simétricas, antes de enviar quaisquer chaves a um POS deve autenticar-se perante o mesmo.	✓			O KDH autentica-se perante P enviando-lhe o seu certificado digital e fazendo prova de posse da respetiva chave privada, por meio de um mecanismo de "desafio-resposta".

Tabela 4: Avaliação do protocolo proposto via ASC X9 TR39

4.8.3 Detalhes de segurança

Nesta secção apresentam-se alguns aspetos relevantes do protocolo do ponto de vista de segurança, de modo a justificar a razão de alguns elementos constarem na sua especificação.

Mecanismo de desafio-resposta

Na fase de autenticação da solução proposta, quando comparada com a fase análoga do protocolo TR34, é percetível a introdução de elementos adicionais não existentes neste último. Além de trocarem os seus certificados de chave pública entre si, P e o KDH trocam também dois valores aleatórios: R_P e R_{KDH}, respetivamente. Estes *nonces* servem como desafio para o outro interlocutor, cuja resposta terá de ser enviada ao proponente fazendo uso da sua chave privada de modo a fazer prova de que está realmente em posse da mesma. Nomeadamente:

- P envia R_P ao KDH para que o assine e devolva essa informação. Caso P valide corretamente a assinatura digital, então P tem garantia de que o KDH está realmente na posse de PR_{signer_F}.
- De igual modo, o KDH envia R_{KDH} a P cifrado pela chave pública deste último. P decifra e envia de volta ao KDH o resultado desta operação. Caso este valor coincida com o original, o KDH garante que é realmente P quem possui PR_P.

Certificate Pinning

Ainda na fase de autenticação foi efetuado outro melhoramento não previsto no documento ASC X9 TR34. Aquando a troca de certificados entre P e o KDH, na solução proposta neste documento é especificado que P, ao receber do KDH o certificado digital $Cert\{PU_{signer\_F}\}$, valida se este é o mesmo certificado que foi previamente injetado pelo seu fabricante, e só nesse caso é que o protocolo prossegue. Este paradigma de apenas confiar em certificados previamente conhecidos e confiáveis denomina-se por *Certificate Pinning* [15].

Entre outras vantagens desta solução, o facto de P confiar apenas num único certificado faz com que não seja necessário o envio de uma lista de certificados revogados (CRL), tal como se efetua no TR34. Em caso de necessidade de revogação ou atualização do par de chaves associados ao certificado $Cert\{PU_{signer\_F}\}$, terá de ser gerado um novo par de chaves no KDH e este terá de garantir a propagação do novo certificado pelos POSs fabricados por F. Este mecanismo de propagação do certificado não se insere no âmbito deste trabalho. No entanto, possíveis soluções poderiam passar por:

- Retirar o POS de serviço para que, em manutenção, lhe fosse injetado o novo certificado.

- O KDH enviar o certificado diretamente aos POSs afetados (desde que a comunicação seja autenticada de modo a que um adversário não tenha a possibilidade de ele próprio induzir em erro o POS para que este instale um certificado associado a um par de chaves sob o controlo do atacante).

Integridade e autenticidade das assinaturas digitais

Um dos problemas existentes relativamente às assinaturas digitais em geral reside no facto de ser possível reassinar um documento com uma chave privada não confiável e, ainda assim, esta assinatura poder ser validada corretamente. Este tipo de ataques, denominados de *Signature Stripping* [8], são possíveis pois tipicamente uma assinatura digital é simplesmente concatenada aos dados assinados, para que outro interveniente possa validar a assinatura. Assim, nada impede a um atacante de intercetar uma mensagem do protocolo proposto, eliminar a assinatura do KDH, reassinar os dados com a sua chave privada e reencaminhar esta nova informação para P. Caso este último tenha sido levado a confiar na chave pública do adversário, validará corretamente as assinaturas deste atacante.

Dado que na parte inicial da fase de transporte foi acordada uma chave de sessão simétrica (K_E) entre o KDH e P (e admitindo que esta chave só é conhecida por estas partes), é possível usá-la para garantir a autenticidade e integridade das assinaturas digitais enviadas por parte do KDH: O KDH pode cifrar as suas assinaturas antes de as enviar a P com a chave efémera K_E. Ao receber estes dados, P decifra a assinatura e caso a valide com sucesso, não só garante que os dados são autênticos e enviados realmente pelo KDH como a assinatura também foi realmente efetuada pelo mesmo (pois só o KDH a poderia ter cifrado com K_E).

Com o *Scyther* é possível reescrever o protocolo de modo a avaliar o impacto de não cifrar as assinaturas digitais. Esta ferramenta indica que, caso esta técnica não fosse posta em prática, não seriam garantidos os requisitos de integridade, autenticação e ordem das mensagens, podendo um atacante alterar *nonces* e reassinar mensagens, quebrando estes requisitos do protocolo. De notar que, ainda assim, não seria quebrada a confidencialidade das chaves a injetar nem da chave de sessão e o atacante teria ainda de conseguir que P confiasse na sua chave pública (o que, tendo em conta a utilização da técnica de *certificate pinning*, seria complexo).

A Figura 25 ilustra o ataque possível caso as assinaturas digitais não sejam cifradas com K_E. A imagem apresenta a linha de execução do protocolo por parte de KDH a azul (denotado por *Alice* na figura) e a verde a execução do protocolo por parte de P (denotado por *Bob*). A laranja é possível visualizar as ações executadas por um atacante, *Eve*. Admitindo a hipótese de P confiar no certificado de *Eve*, o ataque ilustrado apresenta o atacante a recalcular a assinatura digital dos dados enviados pelo KDH (denotado pelas operações de cálculo de síntese criptográfica - *apply* - e cálculo da assinatura digital - *encrypt*) e a reenviá-los a P, bem como a efetuar o reencaminhamento das mensagens

de P para o KDH, perpetuando assim um ataque "homem-no-meio". Adicionalmente, a figura mostra também como seria possível alterar o $nonce$ inicial R_1 para um valor do conhecimento do atacante, $NonceIntruder_1$.

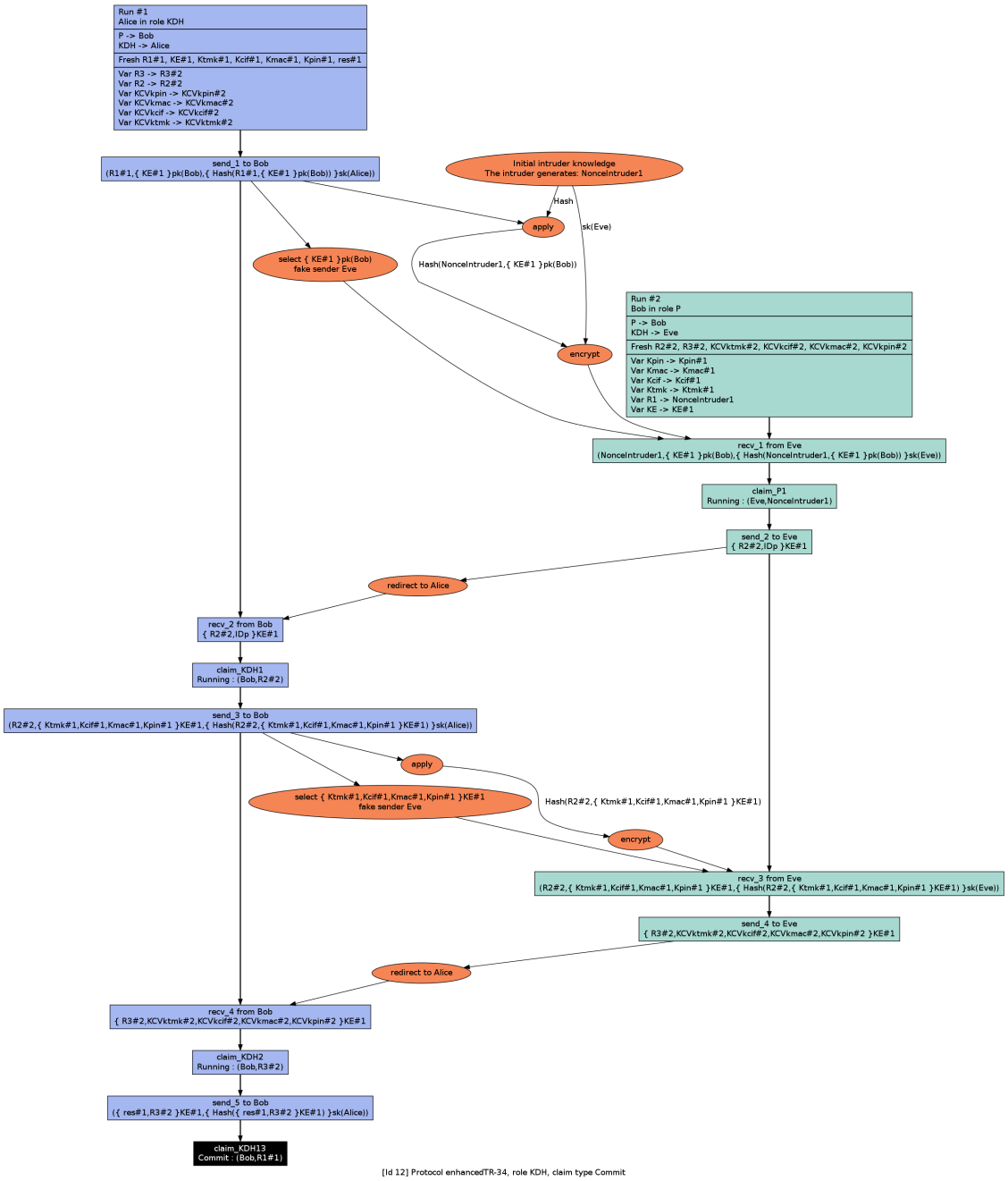

Figura 25: Ataque de *Signature Stripping*, se assinaturas não estiverem cifradas

Medidas contra ataques de criptoanálise

Já na fase final da solução proposta, a razão pela qual se cifra o resultado da execução do algoritmo e um *nonce* R_3 prende-se com a eliminação da possibilidade de se efetuar

um ataque de *Known-plaintext* e tentar, por força bruta, chegar ao valor da chave efémera K_E. Caso este *nonce* fosse inexistente, e admitindo que um adversário tem conhecimento da especificação do protocolo e que sabe a sequência de *bytes* que denota uma notificação de sucesso e de insucesso, este, ao escutar a comunicação, ficaria com conhecimento dos *bytes* que compõem o texto cifrado. Com ambas as informações na sua posse, o adversário pode tentar um ataque por força bruta, tentando todas as combinações possíveis que K_E pode assumir e cifrar o texto correspondente à notificação de sucesso (ou insucesso) com todas essas possibilidades até descobrir a combinação certa. Tendo comprometido a chave de sessão e caso o atacante tenha escutado a restante comunicação executada entre P e o KDH, este poderia comprometer também as chaves do terminal, pois foram transmitidas cifradas por esta chave efémera.

Provas de receção de chaves simétricas

Tal como no TR34, a fase de confirmação do protocolo proposto neste trabalho passa pelo envio dos KCVs das chaves transmitidas. Uma medida adicional não existente nesse protocolo passa pela prova de receção da chave de sessão, K_E, por parte de P. É por esta razão que, na fase de transporte, o KDH só envia a K_{TMK}, K_{CIF}, K_{MAC} e K_{PIN} depois de provar que está em posse da chave de sessão, cifrando o *nonce* R_2 juntamente com o seu número de série de equipamento ID_P. Desta forma, o KDH só envia as chaves simétricas a injetar caso decifre a informação recebida e os dados decifrados contenham ID_P. Note-se que caso este identificador não estivesse presente, o KDH poderia receber quaisquer dados aleatórios cifrados por outra chave qualquer e, ao decifrá-los com K_E, não teria forma de distinguir os *bytes* recebidos do número aleatório real R_2.

É de salientar também que é esta mensagem que corrige, em grande parte, o problema no protocolo original TR34, que recebia este *nonce* em claro que, consequentemente, poderia ser oriundo de um POS fidedigno mas também de um atacante.

4.9 Conclusão do capítulo

Este capítulo descreveu a solução proposta de injeção remota de chaves simétricas em POSs por meio de criptografia assimétrica, desde o momento de pré-inicialização do terminal por parte do fabricante até à sua instalação no local de utilização final, onde recebe o referido material criptográfico simétrico. A proposta, tendo por base o documento ASC X9 TR34, foi avaliada pela ferramenta *Scyther* que mostrou que a solução é robusta em termos da segurança da informação trocada. A solução foi também validada por intermédio do normativo ASC X9 TR39 e essa avaliação revelou-se positiva, mostrando que o protocolo desenhado é praticável, garantindo que está de acordo com as normas em vigor para o âmbito em questão.

Capítulo 5

Conclusão

Este trabalho analisou o "estado da arte" dos atuais métodos para injeção de chaves simétricas em POSs praticados em Portugal, apresentando as suas desvantagens e fraquezas. Após uma investigação sobre possíveis soluções que permitissem descontinuar este carregamento manual atualmente em uso, reuniu-se um conjunto de documentos e normas cujo objetivo é o de definir regras e boas práticas para desenvolver um protocolo de injeção remota de chaves a instalar em POSs. Os mais relevantes foram:

- **ANS X9.24-2:** Com as regras que um protocolo de gestão de chaves simétricas por meio de criptografia assimétrica deve seguir para garantir um alto nível de segurança da informação trocada.
- **ASC X9 TR34:** Com uma sugestão de protocolo de injeção remota de chaves simétricas baseado na norma anterior.
- **ASC X9 TR39:** Com um conjunto de controlos e objetivos que um protocolo de gestão remota de chaves simétricas deve obedecer, servindo de documento de auto-avaliação de qualquer algoritmo desta natureza.

Ao avaliar o protocolo implementado no segundo documento verificou-se que este possui algumas vulnerabilidades que, ainda que não ponham em causa a confidencialidade dos dados, não garantem a autenticidade da informação trocada entre o POS e a entidade distribuidora de chaves simétricas.

Tomando como ponto de partida este documento, foi desenhada e proposta uma solução de injeção remota de chaves simétricas em POSs que corrige os problemas detetados no TR34, adicionando ainda outras medidas de segurança adicionais. Com recurso a uma ferramenta que analisa formalmente protocolos de segurança (*Scyther*) e ao documento de auto-avaliação ASC X9 TR39, conclui-se que a solução proposta satisfaz os requisitos de segurança definidos e que tem potencialidade para ser posto em prática no mundo real, descontinuando os métodos de injeção de chaves atualmente praticados.

A solução proposta permite agilizar o procedimento de inicialização de POSs atualmente em utilização, eliminando significativamente o peso logístico dos mesmos. A

automatização do processo permite também, caso seja necessário, substituir as chaves simétricas destes terminais de forma remota.

O protocolo proposto neste trabalho será apresentado a um fabricante de POSs disponível para trabalhar numa prova de conceito da solução apresentada para que os seus terminais possam receber as chaves criptográficas necessárias de forma remota. Tendo em conta que o *firmware* que atualmente existe nestes terminais não está preparado para executar o protocolo proposto neste documento, este terá de ser modificado para que o POS o possa pôr em prática.

Abreviaturas

3DES Triple Data Encryption Standard
ANSI American National Standards Institute
ASC Accredited Standards Committee
CA Autoridade de Certificação
DEM Data Encryption Mechanism
DES Data Encryption Standard
HSM Hardware Security Module
KBH Key Block Header
KCV Key Check Value
KDH Key Distribution Host
KEM Key Encapsulation Mechanism
KEM-DEM Key Encapsulation Mechanism - Data Encryption Mechanism
KLD Key Loading Device
KRD Key Receiving Device
MAC Message Authentication Code
MIT Massachusetts Institute of Technology
PIN Personal Identification Number
POS Point of Sale
RSA Rivest-Shamir-Adleman
SAM Security Application Module
SPDL Security Protocol Description Language
TDEA Triple Data Encryption Algorithm
TMK Terminal Master Key
TPA Terminal de Pagamento Automático
TPM Trusted Platform Module
TWK Terminal Working Key

Bibliografia

[1] Accredited Standards Committee X9. X9 TR31-2005: Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, 2005.

[2] Accredited Standards Committee X9. ANS X9.24 Part 2:2006, Retail Financial Services. Symmetric Key Management - Using Asymmetric Techniques for the Distribution of Symmetric Keys, 2006.

[3] Accredited Standards Committee X9. X9 TR39-2009: Retail Financial Services Compliance Guideline - Part 1: PIN Security and Key Management, 2009.

[4] Accredited Standards Committee X9. X9 TR34-2012: Interoperable Method for Distribution of Symmetric Keys using Asymmetric Techniques: Part 1 - Using Factoring-Based Public Key Cryptography Unilateral Key Transport, 2012.

[5] Adleman L., Rivest R., Shamir A. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, 1978.

[6] Bellare M., Canetti R., Krawczyk H. Keying Hash Functions for Message Authentication, 1996.

[7] Boeyen S., Cooper D., Farrell S., Housley R., Polk K., Santesson S. RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008.

[8] Caelli W., McCullagh A., Little P. Signature Stripping: A Digital Dillema, 2001.

[9] Cremers, C. Scyther - Semantics and Verification of Security Protocols, 2006.

[10] Cremers, C. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols, 2008.

[11] Cremers, C. Scyther User Manual, 2014.

[12] Diffie W., Hellman M. New Directions in Cryptography, 1976.

[13] Electronic Frontier Foundation (EFF). Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design., 1998.

[14] ENISA. Algorithms, Key Size and Parameters Report – 2013 Recommendations, Version 1.0, 2013.

[15] Evans C., Palmer C., Sleevi R. RFC 7469: Public Key Pinning Extension for HTTP, 2015.

[16] International Organization for Standardization. ISO 13491-1:1998, Banking - Secure Cryptographic Devices (Retail) - Part 1: Concepts, requirements and Evaluation Methods, 1998.

[17] International Organization for Standardization. ISO 11568-2:2012, Financial services - Key management (retail) - Part 2: Symmetric ciphers, their key management and life cycle, 2012.

[18] Kaliski B., Nystrom M. RFC 2986: PKCS #10: Certification Request Syntax Specification, Version 1.7, 2000.

[19] Lowe, G. A Hierarchy of Authentication Specifications, 1997.

[20] Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography, 2001.

[21] Mishra R., Seth S. Comparative Analysis Of Encryption Algorithms For Data Communication, 2011.

[22] Mollin R. An Introduction to Cryptography, 2006.

[23] National Institute of Standards and Technology (NIST). SP 800-12: An Introduction to Computer Security: The NIST Handbook, 1995.

[24] National Institute of Standards and Technology (NIST). Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules, 2001.

[25] National Institute of Standards and Technology (NIST). SP 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005.

[26] National Institute of Standards and Technology (NIST). Federal Information Processing Standards Publication 180-4: Secure Hash Standard (SHS), 2012.

[27] National Institute of Standards and Technology (NIST). SP 800-67 Revision 1: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, 2012.

[28] National Institute of Standards and Technology (NIST). SP 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2012.

[29] National Institute of Standards and Technology (NIST). Glossary of Key Information Security Terms (Revision 2), 2013.

[30] Rivest R. RFC 1321: The MD5 Message-Digest Algorithm, 1992.

[31] Ryan P., Schneider S. Modelling and Analysis of Security Protocols: the CSP Approach, 2001.

[32] Shirey, R. RFC 4949: Internet Security Glossary, Version 2, 2007.

[33] Stallings, W. Cryptography and Network Security - Principles and Practice, 2011.